STATISTICS FOR ALL WALKS ON A LATTICE GRAPH

John Krumm
Microsoft Research
Microsoft Corporation
Redmond, WA 98052
jckrumm@microsoft.com

ABSTRACT

Trajectory data from a moving entity may be handicapped by the temporal gaps between location measurements. We can make inferences about the locations visited during the gaps by postulating all possible paths between pairs of temporally adjacent measurements. These paths are modeled as walks on a spatially discrete grid, represented as a lattice graph. From the collection of possible walks, we can compute statistics about the possible location visits between the measurements, including the probabilities of visiting discrete grid cells and for how long. The paper shows how to compute these statistics, which we share at https://github.com/jckrumm-microsoft/WalksOnLatticeGraph.

***K*eywords** spatial grid · discrete trajectory · lattice graph · walks

1 Introduction

There is value in hypothesizing where a moving entity goes between measurements. We imagine an entity moving through a grid of 1×1 squares, as shown in Figure 1. For any pair of temporally adjacent measurements, we know the corresponding pair of cells and the time it took to go from one to the other. We can hypothesize about the cells that were visited in between. This paper shows how to compute statistics on the in-between cells, assuming the moving entity could take any route on the grid. The statistics show the probability of visiting any cell and the distribution of dwell times in each reachable cell. For geospatial data, these statistics can be used to compute the probability that the moving entity visited a certain location between measurements and the probability distribution of the duration of the visit. This approach has been used for probabilistically interpolating location between measurements in [1].

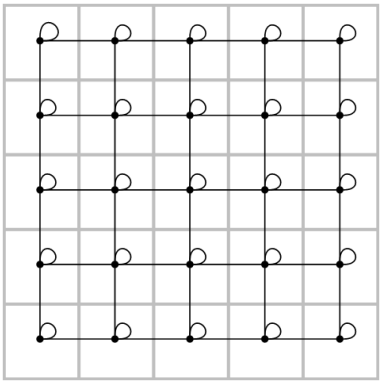


Figure 1: A lattice graph is superimposed on a square grid. Each vertex is shown as a black dot with edges to itself and its 4-connected neighbors.

A convenient way to describe movement through the grid is with a lattice graph. A lattice graph is a graph $G = (V, E)$ whose vertices V are centered on cells in a grid that tile a part of 2D space, an example of which is shown in Figure 1. Since the cells and vertices are inexorably paired, we will use the terms interchangeably. The vertices are denoted as

$v_i \in V$, and the edges are $e_{j,k} \in E$, where $e_{j,k}$ is an undirected edge connecting vertices v_j and v_k. In Figure 1 and the remainder of the chapter, we assume each cell is connected to its 4-connected neighbors, although we could assume 8-connected neighbors as well.

Informally, a trajectory on a lattice graph is a sequence of connected vertices and their connecting edges, i.e. $v_0 e_{0,1} v_1 e_{1,2} v_2 ... e_{T-1,T} v_T$. Omitting the edges as implicit, the trajectory is denoted as $v_0 v_1 v_2 ... v_T$, with the start at v_0, the end at v_T, and positive integer length (or duration) $T \in \mathbb{Z}^+$ indicating the number of edges traversed. We will refer to T as the duration of the trajectory, assuming that the moving entity traverses one edge at every time unit, including possibly the self loop.

In precisely defining trajectories on a graph, we can designate them as *paths* or *walks*[2]. A path consists of a sequence of vertices that are all distinct. Thus it does not cross itself. A walk is more general in that the vertices do not have to be distinct, thus it can cross itself. Because our lattice graph has loop edges on each vertex, the walk can also have an arbitrary number of adjacent, repeated vertices, representing an arbitrary length stay at a vertex.

Because we are looking at trajectories in a spatiotemporal context, such as people traveling around, we introduce time to our walks. Specifically, each walk traverses an edge at every $\Delta T = 1$ time interval. Thus a walk of duration T consists of $T + 1$ vertices, some possibly repeated. We imagine the traversals are instantaneous, with the moving entity spending $\Delta T = 1$ at the new vertex before its next transition. The exception is that the time spent at the first and last vertex is 0.5, which makes it convenient to splice walks together with time spent at interior vertices still 1 and a total duration of T.

Without loss of generality, we assume each walk begins at the vertex at $(x, y) = (0, 0)$ and time $t = 0$. Recall that with grid cells of size 1×1, the coordinates of each cell are $(x, y) \in (\mathbb{Z}, \mathbb{Z})$ on an infinite grid. The last cell/vertex in the walk is at $(x, y) = (X, Y)$. Discretized trajectory data normally gives the starting and ending cells as well as the elapsed time T. Thus we are interested in statistics (e.g. probability of visiting a cell and its dwell time distribution) of all the possible walks from $(x, y, t) = (0, 0, 0)$ to $(x, y, t) = (X, Y, T)$, where t represents discretized time in units of 1.

2 Computing All Walks

The set of all possible walks from $(x, y, t) = (0, 0, 0)$ to $(x, y, t) = (X, Y, T)$ is denoted as $W_{X,Y,T}$. We want to know statistics about this set of walks, specifically the probability of visiting a given cell and the distribution of dwell times in each cell. There appears to be no closed form formulae for these statistics, nor even a closed form giving the number of walks $|W_{X,Y,T}|$. Perhaps the closest solutions are for lattice paths that are more restrictive about the motion from cell to cell and disallow certain regions of the grid [3]. A Mathematics StackExchange question by user DenDenDo suggests using convolutions or adjacency matrices as alternate approaches, but the discussion is not conclusive [4]. There is also a continuous approach called "path integral formulation" based on quantum mechanics that may apply [5].

Given the lack of formulae, we resort to computation. We start with an explicit computation of all walks of a given duration T starting at $(x, y, t) = (0, 0, 0)$. For example, if $T = 2$, we imagine the grid in Figure 2a and its associated 4-connected lattice graph. We denote this set of walks of duration T as W_T. From each vertex on the grid, an entity can move along five different edges, which are the 4-connected neighbors and the self loop. Thus for a walk of duration T, which always traverses T edges, the number of distinct walks is $|W_T| = 5^T$. If we allowed 8-connected neighbors, the number would be 9^T.

For a given T, we do not need an infinite grid. It is adequate to use a grid of size $(2T + 1) \times (2T + 1)$, because the moving entity will only traverse T edges. For example, if $T = 2$, the grid in Figure 2a is adequate.

Our approach is to compute all the walks W_T for a given T and then group by the terminal vertices (X, Y) of each walk in W_T to get the sets $W_{X,Y,T}$. In order to compute W_T (all the walks of length T), we first compute N_T, which is the set of base-5 integers from 0 to $5^T - 1$, as shown in Table 2 for $T = 2$. Each of these 5^T integers will be converted into a distinct walk. The first column in Table 2 shows the 5^2 two-digit integers from 0 to 44 in base five. For the walk represented by each row, the two $(\Delta x, \Delta y)$ moves come from looking up the move corresponding to each of the two digits. Since each walk begins at $(0, 0)$, the location of the terminal vertex is the vector sum of the two moves. The digit-to-move lookup is given in Table 1.

For a given T, we first build a table in a database of the 5^T base 5 numbers, one number for each row, where each row represents a distinct walk on the lattice graph. We chose to explicitly build this table to take advantage of the database-like architecture of our available computing cluster. The table of digits for $T = 1$ is $D_1 = \{0, 1, 2, 3, 4\}$. The digits for $T = 2$ are $D_2 = D_1 \times D_1$, where $\times$ denotes the Cartesian product. Thus $D2 = \{00, 01, 02, ...42, 43, 44\}$, which is shown in the first column of Table 2. In general, $D_T = D_1 \times D_{T-1}$. In a database, the Cartesian product is often implemented as a cross join between two tables.

(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,0)	(-1,0)	(0,0)	(1,0)	(2,0)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)

(a) Coordinates

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

(b) Vertex indices

Figure 2: These are the vertices of the lattice grid for $T = 2$.

Base 5 Digit	Direction	Δx	Δy	Δ Vertex Index
0	no move	0	0	0
1	north	0	1	$2T + 1$
2	south	0	-1	$-(2T + 1)$
3	east	1	0	1
4	west	-1	0	-1

Table 1: This is the lookup table from a base five digit to a relative move on the four-connected lattice graph.

From the digits tables, we compute vertex tables representing the visited vertices on each walk. There is one vertex table for each T, and each row of each vertex table represents one distinct walk on the lattice graph. Instead of storing the vertex coordinates, e.g. $(1, -1)$, we store an integer index for each visited vertex. For a given T, we have a lattice graph with dimensions $(2T + 1) \times (2T + 1)$, with the origin $(0, 0)$ represented as the center vertex. Figure 2a gives an example for $T = 2$. The vertices are numbered in row-major order, starting with 0 in the lower left. The index of the center vertex is $2T(T + 1)$. Figure 2b shows the vertex indices for $T = 2$. A lattice graph of this size covers every possible vertex that can be visited with T moves. These vertex indices are used purely for internal computations as an efficient way to represent the vertices. In the end, we convert from the indices back into (x, y) coordinates for reporting statistics. Table 1 shows how to quickly compute the vertex indices of a vertex's 4-connected neighbors. I.e., for any vertex index, we can compute the indices of its 4-connected neighbors by simply adding the amount given in the last column of Table 1.

We are ultimately interested in visit statistics given knowledge of the start point (always $(0, 0)$), end point, and elapsed time T. Thus we group each vertex table by its terminal vertex and compute statistics on each group.

Before detailing the statistics, we comment on the size of the digit tables and vertex tables above. For a given T, both tables have 5^T rows, one row for each walk, which grows quickly with T. We were able to compute these tables up to $T = 15$ before encountering storage and computation constraints. In our computation, the tables for $T \in [1, 15]$ took over 800 terabytes to store, which is why we only share the statistics, not the actual walks. It is interesting to compare the size of the tables for $T = \mathcal{T}$ to the accumulated size of the tables for $T = 1...\mathcal{T} - 1$. For a finite geometric series, when $b \neq 1$, we have

$$\sum_{i=0}^{n-1} b^i = \frac{b^n - 1}{b - 1}$$

Base 5 Digits	First $(\Delta x, \Delta y)$	Second $(\Delta x, \Delta y)$	First Vertex	Second Vertex	Third (Terminal) Vertex	Walk Count with this Terminal Vertex
00	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	5
01	(0,0)	(0,1)	(0,0)	(0,0)	(0,1)	2
02	(0,0)	(0,-1)	(0,0)	(0,0)	(0,-1)	2
03	(0,0)	(1,0)	(0,0)	(0,0)	(1,0)	2
04	(0,0)	(-1,0)	(0,0)	(0,0)	(-1,0)	2
10	(0,1)	(0,0)	(0,0)	(0,1)	(0,1)	2
11	(0,1)	(0,1)	(0,0)	(0,1)	(0,2)	1
12	(0,1)	(0,-1)	(0,0)	(0,1)	(0,0)	5
13	(0,1)	(1,0)	(0,0)	(0,1)	(1,1)	2
14	(0,1)	(-1,0)	(0,0)	(0,1)	(-1,1)	2
20	(0,-1)	(0,0)	(0,0)	(0,-1)	(0,-1)	2
21	(0,-1)	(0,1)	(0,0)	(0,-1)	(0,0)	5
22	(0,-1)	(0,-1)	(0,0)	(0,-1)	(0,-2)	1
23	(0,-1)	(1,0)	(0,0)	(0,-1)	(1,-1)	2
24	(0,-1)	(-1,0)	(0,0)	(0,-1)	(-1,-1)	2
30	(1,0)	(0,0)	(0,0)	(1,0)	(1,0)	2
31	(1,0)	(0,1)	(0,0)	(1,0)	(1,1)	2
32	(1,0)	(0,-1)	(0,0)	(1,0)	(1,-1)	2
33	(1,0)	(1,0)	(0,0)	(1,0)	(2,0)	1
34	(1,0)	(-1,0)	(0,0)	(1,0)	(0,0)	5
40	(-1,0)	(0,0)	(0,0)	(-1,0)	(-1,0)	2
41	(-1,0)	(0,1)	(0,0)	(-1,0)	(-1,1)	2
42	(-1,0)	(0,-1)	(0,0)	(-1,0)	(-1,-1)	2
43	(-1,0)	(1,0)	(0,0)	(-1,0)	(0,0)	5
44	(-1,0)	(-1,0)	(0,0)	(-1,0)	(-2,0)	1

Table 2: Each row shows one of the 25 possible walks for $T = 2$. In each row, the two base-5 digits are used to look up the five possible moves for the first and second move $(\Delta x, \Delta y)$, with the digit-to-move lookup table in Table!1.

Thus the accumulated number of walks for $T = 1...\mathcal{T} - 1$ is, for $b = 5$,

$$N(\mathcal{T}-1) = \sum_{T=1}^{\mathcal{T}-1} 5^T \tag{1}$$

$$= \frac{5^{\mathcal{T}}-1}{5-1} - 1 \tag{2}$$

$$= \frac{1}{4}(5^{\mathcal{T}}-5) \tag{3}$$

The ratio of the number of rows for $\mathcal{T}$ (i.e. $5^{\mathcal{T}}$) to the accumulated rows for $T = 1...\mathcal{T} - 1$ is then

$$\alpha_{\mathcal{T}} = \frac{5^{\mathcal{T}}}{N(\mathcal{T}-1)} \tag{4}$$

$$= \frac{4 \cdot 5^{\mathcal{T}}}{5^{\mathcal{T}}-5} \tag{5}$$

$$\approx 4 \tag{6}$$

where the approximation comes from saying $5^{\mathcal{T}} - 5 \approx 5^{\mathcal{T}}$, which becomes more accurate with larger $\mathcal{T}$. As an example, $\alpha_{10} = 4.000002048$. The conclusion is that the number of walks for T is about four times the number of accumulated walks for all smaller values of T. Thus moving to the next larger value of T can represent a significant increase (about four times) in memory and computation.

Because the computational burden grows quickly with T, small efficiencies are quickly overwhelmed. For instance, there are clearly spatial symmetries to exploit in the statistics, leading to a possible four or eight times computational savings. However, $T + 1$ has five times the number of walks of T, and four times the number of walks of all the previous T values, as shown above. Thus $O(1)$ speedups are minimally helpful. Meaningful increases in computational efficiency constitute an interesting research problem.

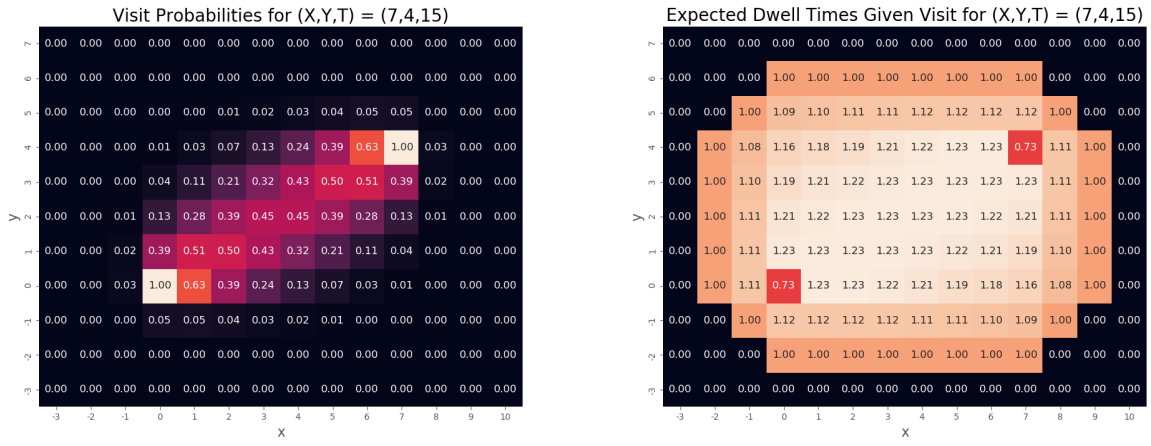
3 Statistics

The vertex tables give the list of vertices visited for each distinct walk, and these are grouped by their terminal vertices to compute $W_{X,Y,T}$, where (X, Y) is the endpoint and T is the walk's duration. The walks themselves are less useful than statistics from the walks. From the constituent walks in $W_{X,Y,T}$, we compute two sets of probabilities. One is the visit probability for each vertex, i.e.

$$P(v|W_{X,Y,T}) = \frac{|W_{X,Y,T}|_v}{|W_{X,Y,T}|} \tag{7}$$

This is the probability of visiting vertex v if the entity went on a walk from the vertex at $(0, 0)$ to the vertex at (X, Y) in time T. The probability is computed in a straightforward way. The numerator $|W_{X,Y,T}|_v$ is the number of walks that pass through vertex v. The denominator $|W_{X,Y,T}|$ is the total number of walks in $W_{X,Y,T}$. The implicit assumption is that each walk in $W_{X,Y,T}$ has equal probability. An interesting extension to this work would be to adjust the probabilities of the individual walks to reflect more realistic behavior. For example, walks that cross themselves may be less probable than those which do not cross themselves, and more direct walks from X to Y may be more probable than those that wander.

As an example, we show $P(v|W_{X,Y,T})$ in Figure 3a. Note that the start and end vertices have a visit probability of 1.0 and that some walks wander outside the bounding box that contains the start and end vertices.



(a) Visit probabilities

(b) Expected dwells given visit

Figure 3: Visit probabilities and expected dwell times given a visit for walks of length $T = 15$ that start at the origin and end at $(7, 4)$.

Another set of probabilities to compute from the raw walks concerns the dwell time in each vertex. Due to the lattice graph's self loops, a walk may dwell in a cell for multiple time steps. A walk can also revisit cells. Both these behaviors lead to varying dwell times among the visited cells. For dwell times, we can easily compute

$$P(v, d|W_{X,Y,T}) = \frac{|W_{X,Y,T}|_{v,d}}{|W_{X,Y,T}|} \tag{8}$$

The probability $P(v, d|W_{X,Y,T})$ is the probability of visiting vertex v and spending time d there. The numerator $|W_{X,Y,T}|_{v,d}$ is the number of walks passing through vertex v that spend time d there. The denominator is the same as in Equation 7.

We can illustrate the dwell probabilities by computing expected dwell times given a visit to a vertex. We have

$$P(v, d|W_{X,Y,T}) = P(d|v, W_{X,Y,T})P(v|W_{X,Y,T})$$

Here, $P(v|W_{X,Y,T})$ is just the visit probability computed in Equation 7. From the above equation, we have

$$P(d|v, W_{X,Y,T}) = \frac{P(v, d|W_{X,Y,T})}{P(v|W_{X,Y,T})}$$

Then the expected dwell time in vertex v, given a visit to vertex v, is

$$\overline{d}|v, W_{X,Y,T} = \sum_d d \cdot P(d|v, W_{X,Y,T})$$

A plot of these expected dwell times for the same walks as in Figure 3a is shown in Figure 3b. Recall that we assume each walk spends a time of 0.5 at the first vertex at the beginning of the walk and 0.5 at the last vertex at the end of the walk. In Figure 3b the expected dwell times at the first and last vertices are both larger than 0.5 (actually 0.73), because some of the walks in $W_{7,4,15}$ loop through and/or stay for a while at the endpoints.

These expected dwell times given a visit are mostly for illustration to demonstrate how to manipulate the probabilities and to show that the resulting dwell times are reasonable. However, if we knew that a certain vertex were visited (i.e. "given a visit"), then we would construct a set of walks that terminate at that given vertex instead of speculating based on the next known vertex.

Another illustration of visit probabilities and expected dwell times appears in Figure 4. All these walks start and end at the origin with $T = 15$. This shows how the walks account for the possibility of moving away from what appears to be a static position.

Statistics for visits and dwells for $T \in [1, 15]$ are available at `https://github.com/jckrumm-microsoft/WalksOnLatticeGraph`.

4 Summary

This paper shows how to compute all walks of a given duration on a lattice graph. This can be useful for speculating about how an entity has moved between location measurements. From the walks, we computed visit probabilities and dwell distributions for the vertices, sharing the results publicly. For longer walks, the computational burden becomes challenging, inviting research for more efficient approaches.

References

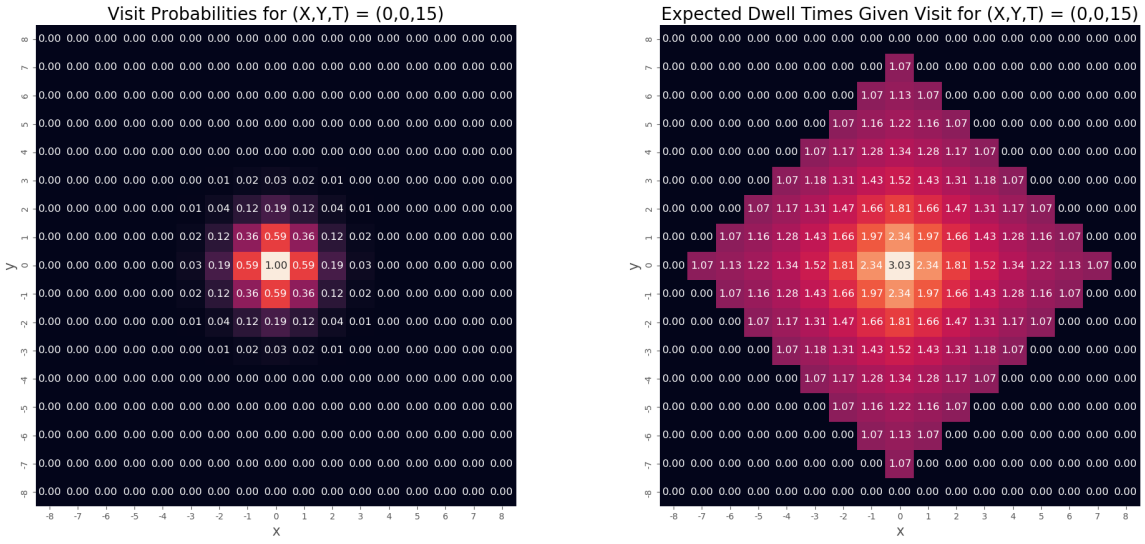
[1] John Krumm. Maximum entropy bridgelets for trajectory completion. In *Proceedings of the 30th ACM SIGSPATIAL international conference on advances in geographic information systems*, Seattle, WA USA, 2022. ACM SIGSPATIAL.

[2] Richard Diestel. *Graph Theory, Fifth Edition*. Springer, Berlin, 2017.

[3] Wikipedia contributors. Lattice path — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Lattice_path&oldid=1094660788`, 2022. [Online; accessed 2-September-2022].

[4] Anupama Aggarwal. Path density between two points. `https://math.stackexchange.com/questions/1691507/path-density-between-two-points`, 2016.

[5] Wikipedia contributors. Path integral formulation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Path_integral_formulation&oldid=1102187594`, 2022. [Online; accessed 25-October-2022].



(a) Visit probabilities

(b) Expected dwells given visit

Figure 4: Visit probabilities and expected dwell times given a visit for walks of length $T = 15$ that start and end at the origin.