Senior Thesis

Yuxuan Chen
Advisor: David Galvin

Contents

Abstract

We study the enumeration problem of self-avoiding walks on the taxi-walk Manhattan lattice, a two-dimensional oriented lattice. The motivation for examining this lattice stems from the problem of estimating a certain value of a parameter in a statistical physics model called the hard-core model. When the parameter attains the very value, this model may transition between order and disorder.

In particular, we focus on a quantity μ_t, called the connective constant of the taxi-walk Manhattan lattice, which governs the rate at which the number of taxi-walks on the lattice grows as the length of the walks increases, and which is also related to the aforementioned parameter of the hard-core model. Previously it had been known that $1.5196 \leq \mu_t \leq 1.5884$. We improve both the upper and the lower bounds, obtaining $1.5572 \leq \mu_t \leq 1.5875$.

1 Introduction

1.1 Self-Avoiding Walks

The topic of self-avoiding walks is elegantly introduced in [1]. A self-avoiding walk is a path visiting any point at most once on a lattice, a common lattice concerning us being $\mathbb{Z}^d$. An interesting problem to look at is the enumeration of such walks. Denote as a_n the number of n-step self-avoiding walks on $\mathbb{Z}^d$ starting at the origin. We have simple bounds for a_n:

$$d^n \leq a_n \leq 2d(2d-1)^{n-1}. \tag{1.1}$$

The lower bound counts the number of walks in which every step is among the d positive coordinate directions, while the upper bound is given by the number of walks devoid of immediate reversals. Let us focus on $\mathbb{Z}^2$. Then, (1.1) becomes $2^n \leq a_n \leq 4 \cdot 3^{n-1}$.

We are interested in the limit

$$\mu = \lim_{n\to\infty} {a_n}^{\frac{1}{n}}, \tag{1.2}$$

which it is easy to prove exists. Since the concatenation of any m-step self-avoiding walk to any n-step self-avoiding walk produces an $(n+m)$-step walk that may not necessarily be self-avoiding, we have

$$a_{n+m} \leq a_n a_m. \tag{1.3}$$

Taking logarithms on both sides of (1.3) gives us a subadditive sequence $\{\log a_i\}$, where

$$\log a_{n+m} \leq \log a_n + \log a_m. \tag{1.4}$$

Here it is necessary to introduce the following lemma, also known as Fekete's Lemma.

Lemma 1.1. For any subadditive sequence $\{c_n\}$, the limit $\lim_{n\to\infty} n^{-1}c_n$ exists in $\mathbb{R}$ and

$$\lim_{n\to\infty} \frac{c_n}{n} = \inf \frac{c_n}{n}.$$

Hence, the limit $\lim_{n\to\infty} n^{-1} \log a_n = \lim_{n\to\infty} \log a_n^{1/n}$ exists and μ, often known as the *connective constant* of the lattice, thus exists. The connective constant is helpful in measuring the growth rate of the number of n-step self-avoiding walks with respect to n. By Lemma 1.1,

$$\begin{aligned} \log \mu &= \inf \frac{\log a_n}{n}, \\ \mu &\leq {a_n}^{\frac{1}{n}}. \end{aligned} \tag{1.5}$$

${a_n}^{\frac{1}{n}}$ is thus an upper bound for μ.

We now look at a lower bound for the connective constant, for which it is worth introducing the concept of *bridges*.

Definition 1.1. An n-step *bridge* ω in $\mathbb{Z}^2$ is an n-step self-avoiding walk satisfying

$$\omega(0) < \omega(i) \leq \omega(n)$$

for $1 \leq i \leq n$, if we denote as $\omega(j)$ the x-coordinate of the j-th vertex of ω. The number of n-step bridges starting at the origin is denoted b_n. By convention, $b_0 = 1$.

A good way to visualize a bridge is to think of a walk that starts by taking a step to the right, never returns to the y-axis, and ends at a point which has the maximum x-coordinate among all the points visited. This final point does not have to be the unique point with maximum x-coordinate. Figure 1 shows an example of a bridge.

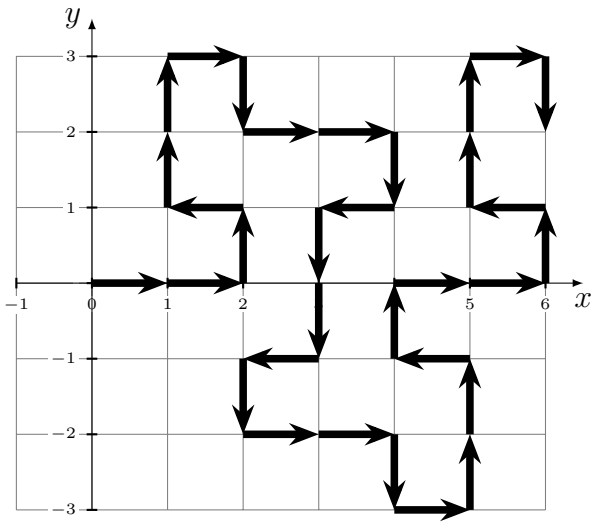

Figure 1: An example of a 32-step bridge starting at the origin in $\mathbb{Z}^2$

Since the concatenation of two bridges always yields a new bridge,

$$\begin{aligned} b_n b_m &\leq b_{n+m}, \\ -\log b_{n+m} &\leq (-\log b_n) + (-\log b_m). \end{aligned} \tag{1.6}$$

Therefore, $\{-\log b_n\}$ is a subadditive sequence and by Lemma 1.1,

$$\begin{aligned} \lim_{n\to\infty} \frac{-\log b_n}{n} &= \inf \frac{-\log b_n}{n}, \\ \lim_{n\to\infty} b_n{}^{\frac{1}{n}} &= \sup b_n{}^{\frac{1}{n}}. \end{aligned} \tag{1.7}$$

Since bridges are essentially self-avoiding walks with extra constraints, we have $0 \leq b_n \leq a_n$, and

$$b_n{}^{\frac{1}{n}} \leq \lim_{n\to\infty} b_n{}^{\frac{1}{n}} \leq \lim_{n\to\infty} a_n{}^{\frac{1}{n}} = \mu. \tag{1.8}$$

Hence, b_n is a lower bound for μ.

1.2 Manhattan Lattice and Taxi-Walks

This paper focuses on the Manhattan lattice, denoted as $\overrightarrow{\mathbb{Z}}^2$, which is a directed lattice on $\mathbb{Z}^2$ simulating the Manhattan Island of New York City whose streets alternate directions. Without loss of generality, we stipulate that in $\overrightarrow{\mathbb{Z}}^2$, the directions on $x = 2k$ ($k \in \mathbb{Z}$) are the positive direction of the y-axis (north), and those on $x = 2k + 1$ the negative direction of the y-axis (south). The directions on $y = 2k$ are the positive direction of the x-axis (east), and those on $y = 2k + 1$ the negative direction of the x-axis (west). The configuration can be visualized in Figure 2.

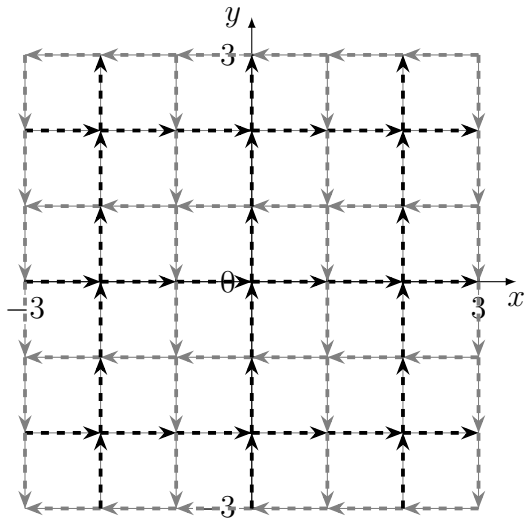

Figure 2: The directions of the Manhattan lattice near the origin

Self-avoiding walks on the Manhattan lattice are similarly defined as those on $\mathbb{Z}^2$ except for directional constraints. We also introduce the concept of *taxi-walks* below.

Definition 1.2. An n-step self-avoiding walk in $\overrightarrow{\mathbb{Z}}^2$ is a sequence $v_0, v_1, ..., v_n$ of distinct vertices with an edge of $\overrightarrow{\mathbb{Z}}^2$ oriented from v_{i-1} to v_i $\forall i = 1, ..., n$. The walk *goes straight* at v_i ($1 \leq i \leq n - 1$) if edges $v_{i-1}v_i$ and v_iv_{i+1} are parallel, and *turns* if the two consecutive edges are perpendicular. A *taxi-walk* is a self-avoiding walk in $\overrightarrow{\mathbb{Z}}^2$ that does not turn at two consecutive vertices.

As an illustration, walks a and b in Figure 3 are an example and a nonexample of taxi-walks starting at the origin, respectively. Walk b is not a valid taxi-walk because it makes two consecutive turns at $(5, -2)$ and $(5, -3)$.

Self-avoiding walks on the Manhattan lattice have been studied for almost as long as self-avoiding walks on $\mathbb{Z}^2$ (see [2] for example), and the connective constant is known to be very close to 1.73. The notion of taxi-walks, on the other hand, was introduced fairly recently in [3].

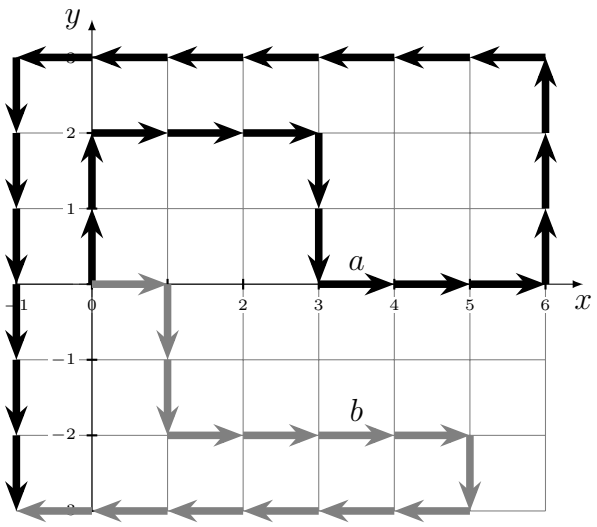

Figure 3: An example and a nonexample of taxi-walks on the Manhattan lattice

The incentive for studying taxi-walks originates from the study of the hard-core or independent set model on $\mathbb{Z}^2$. The problem is essentially the following. An independent set in the two-dimensional grid is a set of pairwise non-adjacent vertices, and it is intended to model a set of massive particles (as opposed to point-masses) occupying space — the fact that the particles are massive means that if a particle is occupying a site then there is no room for any of the sites adjacent to that particle to be occupied simultaneously, hence the introduction of independent sets.

The model has a "density" parameter $\lambda > 0$ that controls the density of a typical configuration of particles. Specifically, each possible configuration I of occupied sites occurs with probability proportional to $\lambda^{|I|}$. Given that there are infinitely many legal configurations, to make this precise it is necessary to use the notion of a Gibbs measure from statistical physics; a full explanation is thus beyond the scope of this thesis. The intuition we are about to delineate instead makes sense if we think of the lattice simply as a very large finite box.

The intuition is that if λ is large, then the probability rule favors the selection of independent sets that are large. One obvious approach to obtain a large independent set is to select a subset of one of the two checkerboard (diagonal) subgrids of the grid. This suggests that when λ is large, a typical configuration from this model is highly structured, consisting of a subset of one of the diagonal subgrids.

On the other hand, if λ is small, then the probability rule is favoring the selection of independent sets that are small, and thus are likely to be fairly disordered.

A longstanding conjecture in statistical physics claims that there is a critical value λ_c such that if $\lambda < \lambda_c$ then the typical configuration from the model is disordered, but that as soon as $\lambda > \lambda_c$ the typical configuration becomes very ordered. In other words, there is a considerably sharp transition between disorder and order, as can be seen in Figure 4. The description we have given is vague, but it can be made perfectly mathematically precise.

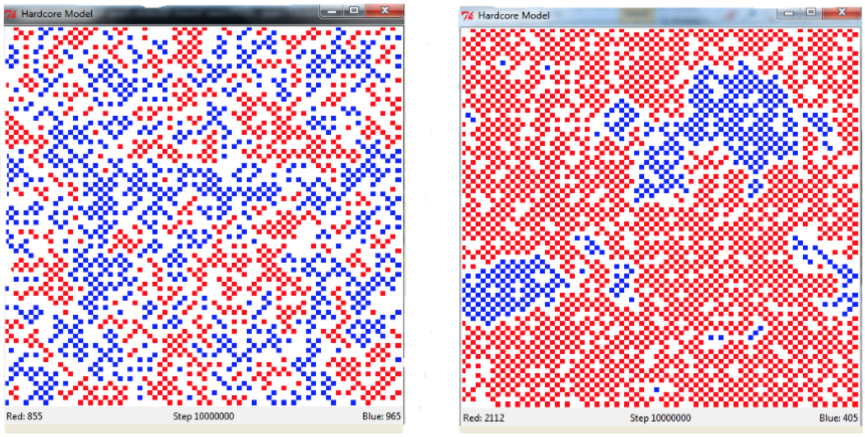

Figure 4: Disorder and order in the hard-core model (simulations by Justin Hilyard)

Simulations have suggested that this critical value λ_c is approximately 3.796, but rigorous results are difficult to prove. In one direction, it is known that if $\lambda < 2.48$ then the model typically selects a disordered set [4]. This is the culmination of a long series of papers.

Very little had been known in the other direction until recently, when in [3] it was shown that if $\lambda > 5.3646$ then the model typically selects an ordered set. Part of the proof worked by showing that if a configuration is disordered, then it has some clustered parts that are predominantly consisting of occupied vertices from one of the two diagonal subgrids, as well as other clustered parts that are predominantly from the other diagonal subgrids. Then there is some unoccupied "contour" separating these two occupied regions; this is costly and so unlikely if λ is large. The gist of [3] is to establish a strong connection between the shape of these "contours" and closed, self-avoiding taxi-walks in the Manhattan lattice. That connection led to the following theorem.

Theorem 1.2. Let μ_t be the connective constant for taxi-walks on $\overrightarrow{\mathbb{Z}}^2$. Then whenever λ satisfies $\lambda > \lambda' = \mu_t^4 - 1$, the typical configuration from the hard-core model on $\mathbb{Z}^2$ is ordered.

This means that any upper bounds we can put on μ_t immediately translate into lower bounds on the phase transition point of the hard-core model, or lower bounds on the start of the interval in which the model is ordered. This theorem is the main reason for our interest in the Manhattan lattice and taxi-walks.

Without loss of generality, we look at taxi-walks starting at the origin of $\overrightarrow{\mathbb{Z}}^2$. Walks starting north and those starting east are symmetric. Denote as $\widetilde{a}_n$ the number of taxi-walks starting east at the origin. Then, the total number of taxi-walks starting at the origin in $\overrightarrow{\mathbb{Z}}^2$ is $2\widetilde{a}_n$. The connective constant μ is defined similarly as that in $\mathbb{Z}^2$, namely,

$$\mu_t = \lim_{n\to\infty} (2\widetilde{a}_n)^{\frac{1}{n}} = \lim_{n\to\infty} (\widetilde{a}_n)^{\frac{1}{n}}, \tag{1.9}$$

which has easy upper and lower bounds:

$$\sqrt{2} \leq \mu_t \leq \frac{1+\sqrt{5}}{2}. \tag{1.10}$$

Since taxi-walks that always take two steps north or two steps east at a time undercounts the number of n-step taxi-walks, we have $2^{n/2} \leq \widetilde{a}_n$ if n is even and $2^{(n+1)/2} \leq \widetilde{a}_n$ otherwise. Hence, $\sqrt{2}$ is a lower bound for μ_t. The upper bound for $\widetilde{a}_n$ is proved below.

Proof. A taxi-walk $v_0, v_1, ..., v_n$ with v_0 as the origin can be encoded by a pair (α, σ), where $\alpha \in \{N, E\}$ and σ a sequence of length $n-1$ over the alphabet $\{s, t\}$. $\alpha = N$ if $v_1 = (0, 1)$ and $\alpha = E$ if $v_1 = (1, 0)$. The i-th entry of σ is s if the walk goes straight and t if the walk turns at v_i, $1 \leq i \leq n-1$. Distinct taxi-walks must have distinct encodings. The number of sequences of length $n-1$ over alphabet $\{s, t\}$ devoid of consecutive occurrences of t is exactly the $(n+1)$-st Fibonacci number f_{n+1} (defined by $f_0 = 0, f_1 = 1,$ and $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$). These f_{n+1} sequences encode taxi-walks as well as directional walks that avoid consecutive turns but may not be self-avoiding, so $\widetilde{a}_n \leq f_{n+1} = O(\frac{1+\sqrt{5}}{2})^n$, and $\mu_t \leq \frac{1+\sqrt{5}}{2}$. □

2 Objectives

The purpose of this thesis to examine methods that have been applied to estimate the connective constants for lattices, and to tailor them for approximation of the connective constant of taxi-walks on the Manhattan lattice.

In Section 3, we study the Goulden-Jackson cluster method, a powerful device for enumerating words over an alphabet that avoid any instance of a specified collection of consecutive substrings. We discuss Alm's method in Section 4, which also helps finding an upper bound but which did not give better results than the Goulden-Jackson method. In Section 5, we report our results applying this method to taxi-walks to get the upper bound for the connective constant. Then in Section 6, we consider an adaptation of the notion of bridges to the taxi-walk setting, and also explain an idea of Kesten's that improves the lower bound of the connective constant by introducing irreducible bridges. We report our results using this approach in Section 7. Some future work is then suggested in Section 8. We also provide in the appendices the pieces of code that we used to implement the Goulden-Jackson method and Kesten's method.

3 Goulden-Jackson Cluster Method

3.1 The Theory behind Goulden-Jackson Method

To find a better upper bound for the connective constant of taxi-walks in the Manhattan lattice, we look at the Goulden-Jackson cluster method. The key of the method is to answer the question: how many possible sequences over a given alphabet are there that do not

contain certain any instance of a given list of subsequences? Our treatment of the answer is based on a description given in a paper by Noonan in [5].

Definition 3.1. For any sequence a, denote the length of a as $\lambda(a)$ and let $wt(a) = s^{\lambda(a)}$. If S is a set of sequences, then $wt(S) = \sum_{a \in S} wt(a)$. Let $W^{(d)}$ denote the set of all sequences on alphabet Σ of size $|\Sigma| = d$. Let $W_i^{(d)}$ be the set of all sequences ending with i ($i \in \Sigma$). Then $W_i^{(d)} = W^{(d)}i$.

Because a sequence is either empty or ends with a letter in the alphabet, we have

$$W^{(d)} = \{[]\} \cup \bigcup_{i \in \Sigma} W^{(d)}i, \tag{3.1}$$

and

$$wt(W^{(d)}) = wt(\{[]\}) + \sum_{i \in \Sigma} wt(W^{(d)}i) = 1 + ds \cdot wt(W^{(d)}). \tag{3.2}$$

Thus, we have

$$wt(W^{(d)}) = \frac{1}{1 - ds}. \tag{3.3}$$

It is necessary here to clarify the concept of generating functions.

Definition 3.2. A generating function $f(s)$ is a formal power series

$$f(s) = \sum_{n=0}^{\infty} p_n s^n$$

whose coefficients are the sequence $\{p_0, p_1, \ldots\}$.

In this case, (3.3) is thus the generating function of $W^{(d)}$, the set of all sequences on alphabet of size d. The k-th term of this generating function is the number of sequences of length k over alphabet of size d. Since $1/(1-ds) = 1 + ds + d^2s^2 + d^3s^3 + \ldots$, it follows that there are d^k sequences of length k over alphabet of size d. This is trivial and we could have calculated this without a generating function. Yet this was just an example to illustrate the more sophisticated application of the method to which we are about to move on. Now we will introduce the concept of *mistakes*.

Definition 3.3. A *mistake* is a given subsequence of consecutive terms that is to be avoided.

In the context of self-avoiding walks in $\mathbb{Z}^2$, if we encode walks as sequences each of whose letters represents the direction taken at each step, then we could treat as mistakes walks that begin and end at the same point, otherwise self-avoiding. Mistakes in the context of taxi-walks in the Manhattan lattice $\overrightarrow{\mathbb{Z}}^2$ could be a superset of the aforementioned set of walks while also including walks that have two consecutive turns.

We continue to use the encoding standard for taxi-walks as stated in the proof of (1.10), but we are making a minor modification. Since our focus is on walks starting east, we will

from now on ignore the capital letter that indicates the starting direction, regarding $\{s,t,s,t\}$ as the same as $(E,\{s,t,s,t\})$ for example. Then the walk $\{t,t\}$ is a mistake because it has two consecutive turns.

Then $\{s,t,s,s,t,s,s,t,s,s,t\}$ would be another mistake as it goes back to the origin in the end, as illustrated in Figure 5. In the ensuing paragraphs, we will keep using these two examples of mistakes above for demonstration. Since every taxi-walk has a unique encoding, counting the number of taxi-walks would amount to counting sequences that avoid a prescribed set of mistakes, which is exactly what the Goulden-Jackson method does.

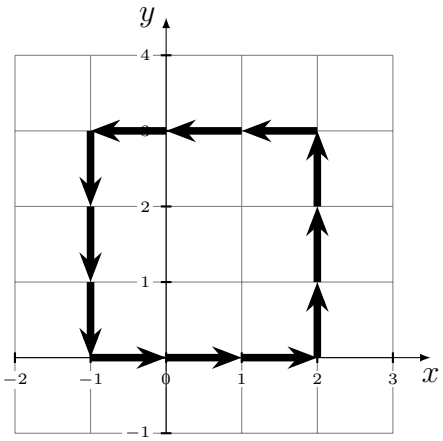

Figure 5: Mistake $\{s,t,s,s,t,s,s,t,s,s,t\}$ on the Manhattan lattice

Before we dive into the details of the Goulden-Jackson cluster method, we need to define a few concepts.

Definition 3.4. A *marked sequence* is a sequence in which a subset of its mistakes is marked.

For example, all of the following are marked sequences. m_1 also counts because an empty subset of its mistakes is marked.

$$\begin{aligned} m_1 &= s,t,s,s,t,s,s,t,s,s,t,t,s \\ m_2 &= \overline{s,t,s,s,t,s,s,t,s,s,t},t,s \\ m_3 &= s,t,s,s,t,s,s,t,s,s,\underline{t,t},s \\ m_4 &= \overline{s,t,s,s,t,s,s,t,s,s,\underline{t,}\ \underline{t},s} \end{aligned}$$

Definition 3.5. Let $\overline{W}^{(d)}$ denote the set of all sequences over alphabet of size d with mistakes marked. So for taxi-walks in For any sequence $\sigma \in \overline{W}^{(d)}$, let $\gamma(\sigma)$ denote the number of mistakes of σ that are marked. Let $\overline{wt}(\sigma) = (-1)^{\gamma(\sigma)} s^{\lambda(\sigma)}$, $\lambda(\sigma)$ still denoting the length of σ. And for a set of sequences S, let $\overline{wt}(S) = \sum_{\sigma \in S} \overline{wt}(\sigma)$.

In our previous examples, $\gamma(m_1) = 0, \gamma(m_2) = \gamma(m_3) = 1$, and $\gamma(m_4) = 2$; $\overline{wt}(m_2) = \overline{wt}(m_3) = -s^{13}$, and $\overline{wt}(m_1) = \overline{wt}(m_4) = s^{13}$. Noonan showed in [5] the following lemma.

Lemma 3.1. The generating function for mistake-avoiding sequences on alphabet of size d is $\overline{wt}(\overline{W}^{(d)})$, which, equivalently, is also the generating function for self-avoiding walks each of whose steps has d possible moves.

Proof. If a sequence $\sigma \in W^{(d)}$ has no mistakes, then σ only appears once in $\overline{W}^{(d)}$ and contributes $s^{\lambda(\sigma)}$ to $\overline{wt}(\overline{W}^{(d)})$.
If σ has k mistakes ($k \geq 1$), then σ contributes $(-1)^r \binom{k}{r} s^{\lambda(\sigma)}$ to $\overline{wt}(\overline{W}^{(d)})$ for $0 \leq r \leq k$, because in $\overline{W}^{(d)}$ there are $\binom{k}{r}$ marked sequences based on σ each with r of σ's k mistakes marked. We know that by the binomial theorem,

$$\sum_{r=0}^{k} (-1)^r \binom{k}{r} = \sum_{r=0}^{k} \binom{k}{r} (-1)^r 1^{k-r} = (-1+1)^k = 0,$$

so σ has no contribution to $\overline{wt}(\overline{W}^{(d)})$ if it has mistakes.
Hence, $\overline{wt}(\overline{W}^{(d)})$ is the generating function of sequences devoid of mistakes. The coefficient of the l-th term of the generating function is exactly the number of mistake-avoiding sequences of length l. □

It is thus critical to figure out what $\overline{wt}(\overline{W}^{(d)})$ is exactly, but before we do that, it is necessary to introduce the concept of *L-clusters*.

Definition 3.6. An *L-cluster* of mistakes is a marked sequence with the following properties:

a) every letter contributes to at least one mistake;

b) its mistakes are fully overlapping.

We denote all L-clusters on alphabet of size d as $L^{(d)}$.

The following are both examples of L-clusters:

$$l_1 = \overline{s,t,s,s,t,s,\underline{s,t,s,s,t},s,s,t,s,s,t}$$
$$l_2 = \overline{\text{s,t,s,}\ \text{s,t,s,}\ \text{s,t,s,s,t,}\ \text{s,s,t,}\ \text{s,s,t}}$$

We can see that l_1 and l_2 are based on the same sequence, but l_2 marks out one more mistake than l_1. Noonan mistakenly stated in [5] that an L-cluster must have all its mistakes marked. He went on to partition $\overline{W}^{(d)}$, in the spirit of (3.1), into the empty sequence, sequences ending with a letter that is not part of a marked mistake, and sequences ending with an L-cluster, since each nonempty marked sequence in $\overline{W}^{(d)}$ either terminates with a marked mistake or does not. The condition that an L-cluster has to have all its mistakes fully marked is too stringent, ignoring marked sequences that may end with subsequences such as l_1 that are not fully marked, and is thus false.

With the partition above in mind, we have

$$\overline{W}^{(d)} = \{[]\} \cup \bigcup_{i \in \Sigma} \overline{W}^{(d)} i \cup \overline{W}^{(d)} L^{(d)}, \tag{3.4}$$

and

$$\overline{wt}(\overline{W}^{(d)}) = 1 + ds\overline{wt}(\overline{W}^{(d)}) + \overline{wt}(\overline{W}^{(d)})\overline{wt}(L^{(d)}). \tag{3.5}$$

Therefore, the generating function for mistake-avoiding sequences on alphabet of size d is

$$\overline{wt}(\overline{W}^{(d)}) = \frac{1}{1 - ds - \overline{wt}(L^{(d)})}. \tag{3.6}$$

So if we want to compute the generating function for self-avoiding walks, we need to compute $\overline{wt}(L^{(d)})$. To calculate this, we segregate $L^{(d)}$ into sets with common terminating mistakes.

Definition 3.7. Let $S[m]$ denote the set of all L-clusters which terminate with the marked mistake m. We have

$$\overline{wt}(L^{(d)}) = \sum_{\text{mistakes } m} \overline{wt}(S[m]). \tag{3.7}$$

Before we move on to one of the most important discussions of this paper, we have to clarify the definitions of the prefix and suffix of a mistake.

Definition 3.8. The prefix of length k of a mistake m is the first k steps of m; the suffix of length k of m is the last k steps of m.

Suppose C is an L-cluster in $S[m]$, then either $C = m$ or $C = Ds$ such that s is a suffix of m and D is an L-cluster in $S[m']$ where m' and m overlap. Then we have a system of equations across different mistakes m, by solving which we can obtain the generating function for mistake-avoiding sequences and equivalently for self-avoiding walks:

$$\overline{wt}(S[m]) = \overline{wt}(m) + \sum_{p,\ p \text{ is a prefix of } m} \left(\sum_{m',\ p \text{ is a suffix of } m'} (-1) s^{\lambda(m) - \lambda(p)} \overline{wt}(S[m']) \right). \tag{3.8}$$

Solving for the $\overline{wt}(S[m])$ for each mistake m and summing them according to (3.7) would allow us to get $\overline{wt}(L^{(d)})$, plugging which in (3.6) will give us $\overline{wt}(\overline{W}^{(d)})$. The issue is that the given set of mistakes may be infinite, and this is true for taxi-walks in the Manhattan lattice. What we can do is to simply use a finite subset of mistakes, and then we would be overcounting the number of self-avoiding walks, or taxi-walks in this case, because we would count walks which include mistakes that should have been in our mentioned subset of mistakes. We will thus be getting an upper bound for μ_t, the connective constant of the taxi-walk Manhattan lattice or the growth rate of the number of taxi-walks with respect to their length.

3.2 Implementation

Instead of calculating the exact generating function in (3.6), we can calculate the coefficients of the Taylor expansion of the generating function inductively from lower to higher terms. Before we describe this inductive relation, bear in mind that the coefficient of the s^k term, as shown in Lemma 3.1, is an exact enumeration of taxi-walks that are self-avoiding up to the first k steps.

For any $\overline{wt}(S[m])$, we iteratively calculate the coefficients of its Taylor expansion from the lower terms. Denote $\overline{wt}(S[m])_k$ as the coefficient of s^k of $\overline{wt}(S[m])$ up to s^k. Suppose we have computed $\overline{wt}(S[m])_j$ for every $0 < j < k$ and for every mistake m. Then,

$$\overline{wt}(S[m])_k = \begin{cases} 0 & \text{for } k < \lambda(m) \\ -1 & \text{for } k = \lambda(m) \\ -\sum_{p,\ p \text{ is a prefix of } m} \left(\sum_{m',\ p \text{ is a suffix of } m'} \overline{wt}(S[m'])_{k-\lambda(m)+\lambda(p)} \right) & \text{for } k > \lambda(m) \end{cases} \tag{3.9}$$

We modeled after the Maple code provided in [5] and implemented the Goulden-Jackson method with the Python code shown in A.1, utilizing the above inductive relation. Since the number of mistakes fed into the Goulden-Jackson method has to be finite, the generating function of the taxi-walks must be rational. Consequently, the coefficients of the Taylor expansion of the very generating function are asymptotic to a constant times α^n where α is the smallest positive root of the denominator of the generating function. Thus, the n-th root of the coefficient converges to α, which is the connective constant. And we will get an upper bound on the connective constant if we look at a large enough n.

4 Alm's Method

We also attempted Alm's method in [6] to obtain a tighter upper bound for the connective constant of the taxi-walks, but the approach requires unfeasibly large computing power for matrix construction. The previous best result for the upper bound obtained in [3] using Alm's method already tried to construct a 10057×10057 matrix, and if we were to make any incremental improvement, we would have to create a significantly larger matrix and the computing power we had did not allow us to do that within a limited amount of time, so eventually we did not go down this path. Nonetheless, we explain the gist of Alm's method below as it is a remarkably ingenious mechanism.

Consider a finitely generated lattice such as $\mathbb{Z}^2$ or $\overrightarrow{\mathbb{Z}^2}$. Let $\mathbf{G}(m, n)$ be the matrix whose row indices represent all different m-step self-avoiding sub-walks that any n-step self-avoiding walk may start with, and whose column indices represent all different m-step self-avoiding sub-walks that any n-step self-avoiding walk may end with, where $m < n$. Two m-step self-avoiding walks are considered equivalent if one can be mapped on the other by translation, rotation, or reflection. Each entry $g_{ij}(m, n)$ of the matrix $\mathbf{G}(m, n)$ is the number of n-step

self-avoiding walks that start with the m-step self-avoiding walk i maps to and end with the m-step self-avoiding walk j maps to. Alm then presented the following theorem.

Theorem 4.1. If μ denotes the connective constant of the lattice and $\lambda_1(\mathbf{A})$ denotes the largest eigenvalue of any lattice $\mathbf{A}$,

$$\mu \leq (\lambda_1(\mathbf{G}(m,n))^{\frac{1}{n-m}}$$

Computing the largest eigenvalue of a matrix is fast. The time-consuming, let alone space-consuming, part is constructing the very matrix. That is why we did not put our focus on Alm's method, even though it is absolutely phenomenal.

5 Results for the Upper Bound

The previous best result for the upper bound was obtained in [3] using Alm's method with $m = 20, n = 60$. The upper bound μ_t was computed to be 1.5884 and $\lambda = \mu_t^4 - 1 = 5.3656$ by Theorem 1.2, which states that the hard-core model exhibits order for all values of $\lambda > \lambda'$. The Goulden-Jackson cluster method was also attempted in [3], but the upper bound obtained was not as good as 1.5884. The limitation was that only mistakes of length up to 40 (there are 317,267 of them) were used.

We generated a list of all mistakes of length up to 48 (there are 8,009,145 of them, see `https://drive.google.com/file/d/0B66P3nZV3KpUYXA2TmNpY2ZrTFk/view?usp=sharing`) using the code presented in Section A.2. Running so many mistakes through the Goulden-Jackson, however, was slow and the best result we obtained ended up using only mistakes of length up to 44 (there are 1,721,327 of them). We used the Goulden-Jackson method to get the number of taxi-walks of length 802 that avoided mistakes of length at most 44, and computed the 802-nd root of that number to get an upper bound on μ_t of 1.587459, and $\lambda' = \mu_t^4 - 1 = 5.350531$.

6 Kesten's Irreducible Bridge Method

6.1 The Theory behind Irreducible Bridges

We now consider lower bounds on the connective constant of taxi-walks. We have already mentioned in Definition 1.1 the notion of bridges and how it can be deployed to obtain a lower bound for the connective constant. Here we define and explain a modification that is suitable for studying the Manhattan lattice — Kesten's theory of irreducible bridges, which significantly improves the lower bounds that can be obtained. Kesten's original treatment for ordinary self-avoiding walks appears in [7]; our treatment, however, follows Alm and Parviainen's presentation in [8]. We begin with defining *bridge taxi-walks* by modifying Definition 1.1.

Definition 6.1. An n-step *bridge taxi-walk* $\overrightarrow{\omega}$ in $\overrightarrow{\mathbb{Z}}^2$ is an n-step self-avoiding walk satisfying

i. $\overrightarrow{\omega}(0) < \overrightarrow{\omega}(i) \leq \overrightarrow{\omega}(n)$ for $1 \leq i < n$, and

ii. $\overrightarrow{\omega}(n-1) < \overrightarrow{\omega}(n)$.

The number of n-step bridge taxi-walks starting at the origin is denoted $\overrightarrow{b_n}$. By convention, $\overrightarrow{b_0} = 1$.

A nice approach to imagine a bridge taxi-walk is to consider a walk that starts by taking a step to the right, never returns to the y-axis, and ends at a point which has the maximum x-coordinate among all the points visited, along with the extra constraint that the last step is a step to the right. The raison d'être of this additional constraint is to avoid consecutive turns in the concatenation of any two bridges. Also, the final vertex of the walk does not need to be the unique vertex with maximum x-coordinate. Figure 6 demonstrates an illustration of a bridge taxi-walk.

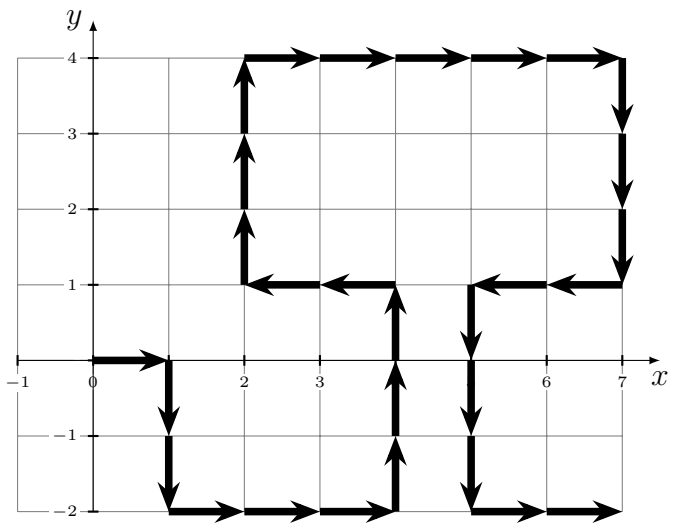

Figure 6: An example of a 29-step bridge taxi-walk in the Manhattan lattice

Let $B(x) = \sum_{n\geq 0} b_n x^n$ be the generating function of the b_n's. We have

$$B(x) = 1 + x + x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + 7x^7 + 11x^8 + 16x^9 + 25x^{10} + \dots \tag{6.1}$$

(see `http://www3.nd.edu/~dgalvin1/TD/index.html`, the computations here due to Blanca in [3]).

To move on, it is necessary to introduce the concept of *irreducible bridges*.

Definition 6.2. An *irreducible bridge* is a bridge taxi-walk $\overrightarrow{\omega}$ that cannot be decomposed into two bridge taxi-walks. Specifically, there does not exist $1 \leq i \leq n$ satisfying

i. $\overrightarrow{\omega}(0) < \overrightarrow{\omega}(j) \leq \overrightarrow{\omega}(i)$ for $1 \leq j < i$,

ii. $\overrightarrow{\omega}(i-1) < \overrightarrow{\omega}(i)$,

iii. $\overrightarrow{\omega}(i) < \overrightarrow{\omega}(k) \leq \overrightarrow{\omega}(n)$ for $i+1 \leq k < n$, and

iv. $\overrightarrow{\omega}(n-1) < \overrightarrow{\omega}(n)$,

Denote by β_n the number of irreducible bridges of length n. By convention, $\beta_0 = 0$.

The taxi-walk shown in Figure 6 is not an irreducible bridge because it can be decomposed into two bridge taxi-walks at vertex $(4,4)$, one from the origin to $(4,4)$ and the other from $(4,4)$ to $(7,-2)$. Both of these bridge taxi-walks, however, are irreducible bridges.

Fix $n \geq 1$. For each $\ell \geq 1$, each solution to $k_1 + \ldots + k_\ell = n$ with each $k_i \geq 1$, and each selection of a collection of irreducible bridge $p_1, \ldots, p_\ell$ with p_i of length k_i for each i, there corresponds a bridge of length n obtained by concatenating the p_i's (with suitable translations and reflections where necessary). Moreover, each bridge of length n is obtained exactly once in this process. It follows that

$$b_n = \sum_{\ell \geq 1} \sum_{k_1 + \ldots + k_\ell = n} \prod_{i=1}^{\ell} \beta_i. \tag{6.2}$$

and, using the general theory of generating function (see Rule 4, Section 2.2 of [9]),

$$B(x) = \frac{1}{1 - \beth(x)}$$

where $\beth(x) = \sum_{n \geq 0} \beta_n x^n$. Therefore, $\beth(x) = 1 - (1/B(x))$. For any $N \geq 1$, the coefficients of the power series of $1/B(x)$ up to the coefficient of x^N are determined by the coefficients of the power series of $B(x)$ up to x^N, and so the coefficients of $\beth(x)$ up to x^N are determined by the coefficients of $B(x)$ up to x^N. We know the coefficients of $B(x)$ up to x^{60} from (6.1), so we also know the coefficients of $\beth(x)$ up to x^{60}. A Mathematica computation yields $\beth(x) = x + x^4 + x^6 + x^8 + x^{10} + x^{12} + x^{13} + x^{14} + 3x^{15} + 3x^{16} + 6x^{17} + 8x^{18} + 12x^{19} + 17x^{20} + 27x^{21} + 33x^{22} + 63x^{23} + 67x^{24} + 144x^{25} + 145x^{26} + 321x^{27} + 326x^{28} + 706x^{29} + 728x^{30} + 1581x^{31} + 1614x^{32} + 3598x^{33} + 3610x^{34} + 8224x^{35} + 8194x^{36} + 18814x^{37} + 18730x^{38} + 43062x^{39} + 42947x^{40} + 98875x^{41} + 98558x^{42} + 228098x^{43} + 226872x^{44} + 528070x^{45} + 524588x^{46} + 1225338x^{47} + 1217757x^{48} + 2849693x^{49} + 2834637x^{50} + 6641524x^{51} + 6613633x^{52} + 15514432x^{53} + 15464004x^{54} + 36317487x^{55} + 36238612x^{56} + 85173845x^{57} + 85103200x^{58} + 200093253x^{59} + 200232649x^{60} + \ldots$.

We used the code in A.3 to verify the above equation up to x^48 and it was a perfect match.

Notice that $\beth(x) = 1$ has a unique positive root $r_{\text{pos}} < 1$. Let r be any upper bound on r_{pos}. Knowing (6.3), the standard theory of generating functions (see Section 2.4 of [9]) tells us that r_{pos} is an upper bound on the radius of convergence of $B(x)$, and so is r. We then have

$$\limsup_{n \to \infty} b_n^{1/n} \geq 1/r.$$

But we also know that $b_n \leq \widetilde{a}_n$ where $\widetilde{a}_n$ is the number of taxi-walks of length n (only considering the ones starting east), so

$$\limsup_{n\to\infty} b_n^{1/n} \leq \limsup_{n\to\infty} \widetilde{a}_n^{1/n} = \mu_t$$

where μ_t is the taxi-walk connective constant. Hence $\mu_t \geq 1/r$.

Henceforth, to get lower bounds on μ_t, we want to get upper bounds on r_{pos}, the unique positive solution to $\beth(x) = 1$.

Consider a sequence $\{\beta'_n\}_{n=1}^{\infty}$ with $0 \leq \beta'_n \leq \beta_n$ for each n. Set $\beth'(x) = \sum_{n=1}^{\infty} \beta'_n x^n$. As is the case with $\beth(x)$, the equation $\beth(x) = 1$ has a unique positive solution. If this solution is r, then we have $r \geq r_{\text{pos}}$. So our conclusion is the following.

Theorem 6.1. Suppose β_n be the number of irreducible bridges of length n in the taxi walk Manhattan lattice. If $0 \leq \beta'_n \leq \beta_n$ for all n, and if $\sum_{n=1}^{\infty} \beta'_n x$, then $\mu_t \geq 1/r$.

This is why it is of value to seek lower bounds on β_n. In the ensuing subsection, we are describing an approach that we used to construct certain subsets of irreducible bridges, thus getting lower bounds.

6.2 "Mountain Ranges" Construction of Irreducible Bridges

Suppose that p is a lattice path, that starts at the origin, takes steps in increments of $(1, 1)$ and $(1, -1)$, ends at the point (m, w) (so takes a total of m steps, and has a net height change of w). Suppose further that p reaches the line $y = w$ at some intermediate point (before the last step), that it returns to the x-axis at some point after the point at which it has reached the line $y = w$, and that it never goes below the x-axis or above the line $y = w$. Assume $w \geq 1$. Let t be the number of vertices at which p turns. Call p an (m, w, t) path. Let the vertices of p be $(v_1, \ldots, v_{m+1})$; note that $v_1 = (0, 0)$, $v_2 = (1, 1)$, $v_m = (m - 1, w - 1)$ and $v_{m+1} = (m, w)$, and that exactly t of the v_i, p turns.

For each integer $\ell \geq 0$, and for each weak composition $c_1 + \ldots + c_m = \ell$ of ℓ into m parts, consider the following walk in the Manhattan lattice.

- Start at $(0, 0)$ and step right.
- Take $2c_1$ steps down, then step right twice.
- If v_2 is not a vertex where p turns, take $2c_2$ steps down, then take a step to the right twice. If v_2 is a vertex where p turns, take $3 + 2c_2$ steps down, then take a step to the left twice.
- If v_3 is not a vertex where p turns, take $2c_3$ steps down, then take a step to the right twice (if p is moving up as it passes through v_3) or a step to the left twice (if p is moving down as it passes through v_3). If v_3 is a vertex where p turns, take $3 + 2c_2$ steps down, then take a step to the left twice (if p is moving down after it passes through v_3) or a step to the right twice (if p is moving up after it passes through v_3).

- Repeat for all vertices of p, up to v_m.

The end result is an irreducible bridge of length

$$2m + 1 + 2\ell + 3t.$$

In fact, there are $\binom{m+\ell-1}{m-1}$ such irreducible taxi-walk bridge of this length, one for each weak composition of ℓ into m parts.

Note that t must always be even, so $2m + 1 + 2\ell + 3t$ must always be odd.

By this scheme, we can get a lower bound on β_n, the number of irreducible bridges of length n (for odd n). Say that an (m, w, t) path p is n-good if $2m + 1 + 3t \leq n$ and if $n - (2m + 1 + 3t)$ is even (this second condition is redundant, since n must be odd and t even). There is a unique choice of $\ell \geq 0$, namely $(n-(2m+1+3t))/2$, for which the process described above produces irreducible bridges of length n, and it produces

$$\binom{m + (n - (2m + 1 + 3t))/2 - 1}{m - 1} = \binom{n/2 - 3t/2 - 3/2}{m - 1} = \binom{(n - 3t - 3)/2}{m - 1}$$

such. So we get

$$\beta_n \geq \sum_{n\text{-good paths}} \binom{(n - 3t - 3)/2}{m - 1}.$$

Here is a scheme for producing n-good paths: consider the path that goes straight from the y-axis up to the line $y = w$, straight back to the x-axis, and then straight back to the line $y = w$. This path has $m = 3w$ and $t = 2$, so it can be thought of as a $(3w, w, 2)$ path. This path will be n-good for $6w + 7 \leq n$. It follows that for each odd $n \geq 13$, we get

$$\beta_n \geq \sum_{1 \leq w \leq (n-7)/6} \binom{(n - 9)/2}{3w - 1}.$$

For example, for $n = 59$ this gives $\beta_n \geq 11184810$, around 5.6% of the true value of 200093253.

Here is a generalization of this idea: consider the path that goes straight from the y-axis up to the line $y = w$, straight back to the x-axis, and straight back to the line $y = w$, and repeats this zig-zagging until it returns to the line $y = w$ for the kth time, for some parameter $k \geq 2$ ($k = 2$ is the case we have just considered). This path has $m = (2k - 1)w$ and $t = 2k - 2$, so it can be thought of as a $((2k - 1)w, w, 2k - 2)$ path. This path will be n-good for $(4k - 2)w + 6k - 5 \leq n$. It follows that for each odd $n \geq 10k - 7$, we get

$$\beta_n \geq \sum_{1 \leq w \leq (n-6k+5)/(4k-2)} \binom{(n - 6k + 3)/2}{(2k - 1)w - 1},$$

and so, aggregating over choices of k, for $n \geq 13$

$$\beta_n \geq \sum_{2 \leq k \leq (n+7)/10} \sum_{1 \leq w \leq (n-6k+5)/(4k-2)} \binom{(n - 6k + 3)/2}{(2k - 1)w - 1}.$$

For example this gives $\beta_{59} \geq 12078275$, around 6% of the true value.

We may improve things more. For example, at each of the $(2k-1)(w-1)$ places where our basic $((2k-1)w, w, 2k-2)$ path does not turn, we may insert any number of "vees": either a down step followed by an up step if the path is going up, or an up step followed by a down step if it is going down. Suppose we insert a such vees; the number of ways to do so is the number of weak compositions of a into $(2k-1)(w-1)$ parts, or $\binom{(2k-1)(w-1)+a-1}{a}$. Each such mode of insertion leads to a $((2k-1)w+2a, w, 2k-2+2a)$ path. This path will be n-good for $(4k-2)w+6k-5+10a \leq n$. This leads to

$$\beta_n \geq \sum_a \sum_k \sum_w \binom{(2k-1)(w-1)+a-1}{a} \binom{(n-6k+3-6a)/2}{(2k-1)w+2a-1}.$$

Where $a \in [0, (n-13)/10]$, $k \in [2, (n+7-10a)/10]$ and $w \in [1, (n-6k+5-10a)/(4k-2)]$ in the sums above.

This improves a_{59} to ≥ 13798649, around 6.89% of the true value.

7 Results for the Lower Bound

Using the "Mountain Ranges" approach and setting $\beta'_n = \beta_n$ for $n \leq 60$, and $\beta_n = 0$ otherwise, we get $\mu_t \geq 1.557012397$ ($\mu_t^4 - 1 \geq 4.87717$). Using instead

$$\beta'_n = \sum_a \sum_k \sum_w \binom{(2k-1)(w-1)+a-1}{a} \binom{(n-6k+3-6a)/2}{(2k-1)w+2a-1}$$

(where $a \in [0, (n-13)/10]$, $k \in [2, (n+7-10a)/10]$ and $w \in [1, (n-6k+5-10a)/(4k-2)]$ in the sums above) for $61 \leq n' \leq 901$, n' odd, improves this to $\mu_t \geq 1.557118239$ ($\mu_t^4 - 1 \geq 4.8787$).

8 Future Work

There is possibility for further improvement. If we have ample computing power, we can compute the exact number of irreducible bridges of lengths 61 and greater.

Another observation is that it seems very clear from the data up to $n = 60$ that $\beta_n \geq 2\beta_{n-2}$ for all large enough n (this has certainly settled down into a clear pattern by $n = 60$). Incorporating this empirical lower bound leads to a jump in connective constant lower bound from 1.557118 to 1.559322. But this would require a strict proof of this inequality.

We may also make use of the following lemma, although there seems to be little improvement on the lower bound. If we can prove $\beta_{n+k} \geq \beta_n$ for a much smaller k, the improvement would be much greater.

Lemma 8.1. $\beta_{n+17} \geq \beta_n$ for all $n \geq 1$.

Proof. For any irreducible bridge starting with a straight, flip the irreducible bridge upside-down, prepend it with a 17-step walk as shown in Figure 7, and still get an irreducible bridge. Otherwise, the irreducible bridge starts with a turn, and we can flip the irreducible bridge upside-down and prepend it with another 17-step walk as shown in Figure 8, and still get an irreducible bridge.

□

References

[1] N. Madras and G. Slade, *The Self-Avoiding Walk*, Springer, 1996.

[2] M. Barber, Asymptotic results for self-avoiding walks on a Manhattan lattice, *Physica* **48** (1970), 237–241.

[3] A. Blanca, D. Galvin, D. Randall and P. Tetali, Phase Coexistence and Slow Mixing for the Hard-Core Model on $\mathbb{Z}^2$, *Lecture Notes in Comput. Sci.* **8096** (*Proc. APPROX-/RANDOM 2013*) (2013), 379—394.

[4] J. Vera, E. Vigoda and Y. Yang, Improved bounds on the hard-core model for phase transition in 2-dimensions, arXiv:1306.0431.

[5] J. Noonan, New Upper Bounds for the Connective Constants of Self-Avoiding Walks, *J. Stat. Phys.* **91**, 871-888, 1998.

[6] S. E. Alm, Upper Bounds for the Connective Constant of Self-Avoiding Walks, *Combin. Probab. Comput.* **2**, 115-136, 1993.

[7] H. Kesten, On the number of self-avoidingwalks, *J. Math. Phys.* **4**, 960-9, 1963.

[8] S. E. Alm and R. Parviainen, Bounds for the connective constant of the hexagonal lattice, *J. Phys. A: Math. Gen.* **37**, 549-560, 2004.

[9] H. S. Wilf, Generatingfunctionology. *Bull. Amer. Math. Soc. (N.S.)* **25**, no. 1, 104-106, 1991.

A

A.1 Goulden-Jackson Cluster Method Python Code

```
import decimal
import math
import sys

if len(sys.argv) != 5 and len(sys.argv) != 4:
  sys.exit("usage: ./gjseries_v2.py <alphabet_size> <input_file> <num_steps> (<
      output_file>)")
a_size = int(sys.argv[1])
f = open(sys.argv[2], "r")
lines = f.readlines()
first = True

lis = []

# get suffices and maximal mistake length
suffi = set()
maxlen = 0
mistakes = []
for i in xrange(len(lines)):
  line = lines[i]
  mistake = eval(line.rstrip()[:-1])
  mistakes.append(mistake)
  if len(mistake) > maxlen:
    maxlen = len(mistake)
  for ind in xrange(1, len(mistake)):
    suffi.add(''.join([str(code) for code in mistake[ind:]]))

efes = [0 for i in xrange(maxlen - 1)]

nuterms = int(sys.argv[3])
lt = {}
c = []
curr_mu = None

for i1 in xrange(nuterms + 1):
  tot = 0
  x = {}

  for i in xrange(len(mistakes)):

    mistake = mistakes[i]
```

```
  gu = 0
  if len(mistake) == i1:
    gu = -1

  for r in xrange(1, len(mistake)):
    pref = mistake[:(len(mistake) - r)]
    pref = ''.join([str(k) for k in pref])
    if pref in suffi:
      if not pref in lt:
        lt[pref] = efes
      gu = gu - lt[pref][maxlen - 1 - r]

  for j in xrange(1, len(mistake)):
    suf = mistake[j:]
    suf = ''.join([str(k) for k in suf])
    if suf not in x:
      x[suf] = 0
    x[suf] += gu

  tot += gu

lis.append(tot)

if i1 == 0:
  c_k = 1 / (1 - tot)
  c.append(c_k)
else:
  c_k = 0
  for i in xrange(i1):
    b = -lis[i1 - i]
    if i == i1 - 1:
      b = -a_size - lis[1]
    c_k += b * c[i]
  c_k = -c_k/(1-lis[0])
  c.append(c_k)

if i1 > 1:
  curr_mu = decimal.Decimal(str(c_k)) ** decimal.Decimal(str(1.0/i1))
  print i1, curr_mu
  ouf = open(sys.argv[4], "w")
  ouf.write(str(i1) + " " + str(curr_mu))
  ouf.close()

for suff in suffi:
  suf = ''.join([str(k) for k in suff])
```

```
    if suf not in lt:
      lt[suf] = efes
    resh = lt[suf]
    if suf not in x:
      x[suf] = 0
    if len(resh) < 2:
      lt[suf] = [x[suf]]
    else:
      lt[suf] = resh[1:]
      lt[suf].append(x[suf])
```

A.2 Mistake Generating Code

```
import sys

def is_polygon(arr):
  coords = [(0,0),(0,1)]
  curr_dir = 'y'
  for i in xrange(len(arr)):
    num = arr[i]
    last = coords[-1]
    if num == 1:
      if curr_dir == 'y':
        new_coord = (last[0], last[1] + 1 - 2 * (last[0] % 2))
      else:
        new_coord = (last[0] + 1 - 2 * (last[1] % 2), last[1])
    else:
      if curr_dir == 'y':
        curr_dir = 'x'
        new_coord = (last[0] + 1 - 2 * (last[1] % 2), last[1])
      else:
        curr_dir = 'y'
        new_coord = (last[0], last[1] + 1 - 2 * (last[0] % 2))
    if new_coord in coords:
      return i == len(arr) - 1 and coords[0] == new_coord
    coords.append(new_coord)
  return False

def dfs(arr, curr_depth, end_depth, extra_stuff):
  if curr_depth > end_depth:
    if is_polygon(arr):
      ouf = extra_stuff[0]
      ouf.write(str(arr) + ",\n")
      print "\r" + ''.join([str(i) for i in arr]),
    return
```

```
  for i in xrange(1,3):
    if len(arr) > 0 and arr[-1] == 2 and i == 2:
      break
    dfs(arr + [i], curr_depth + 1, end_depth, extra_stuff)

if len(sys.argv) != 3:
  sys.exit("usage: ./mistakes_v1.py <mistake_length> <output_file>")
mist_len = int(sys.argv[1])
ouf = open(sys.argv[2], "w")
dfs([], 1, mist_len, [ouf])
ouf.close()
```

A.3 Irreducible Bridge Enumerating Code

```
import numpy
import sys

def is_irred_bridge(arr):
  last = (1,0)
  existing_coords = {(0,0):True, (1,0):True}
  coords = [(0,0), (1,0)]
  curr_dir = 'x'
  for i in xrange(len(arr)):
    num = arr[i]
    if num == 1:
      if curr_dir == 'y':
        new_coord = (last[0], last[1] + 1 - 2 * (last[0] % 2))
      else:
        new_coord = (last[0] + 1 - 2 * (last[1] % 2), last[1])
    else:
      if curr_dir == 'y':
        curr_dir = 'x'
        new_coord = (last[0] + 1 - 2 * (last[1] % 2), last[1])
      else:
        curr_dir = 'y'
        new_coord = (last[0], last[1] + 1 - 2 * (last[0] % 2))
    if new_coord[0] <= 0 or new_coord in existing_coords:
      return False
    last = new_coord
    existing_coords[last] = True
    coords.append(new_coord)
  if curr_dir != 'x' or last[0] < max([x[0] for x in coords]):
    return False
  for i in xrange(1, len(coords) - 1):
    xx = coords[i][0]
```

```
    if coords[i - 1][1] != coords[i][1] or coords[i + 1][1] != coords[i][1]:
      continue
    if max([x[0] for x in coords[:i]]) <= xx and xx < min([x[0] for x in coords[i
       +1:]]):
      return False
  return True

def dfs(arr, curr_depth, end_depth):
  global count, mu
  if curr_depth > end_depth:
    if is_irred_bridge(arr):
      count += 1
      print "Lower bound:", "{0:.6f}".format(round(mu,6)), "; step", end_depth,
          ''.join([str(x) for x in arr]), "\r",
    return
  for i in xrange(1,3):
    if len(arr) > 0 and arr[-1] == 2 and i == 2:
      break
    dfs(arr + [i], curr_depth + 1, end_depth)

if len(sys.argv) != 3:
  sys.exit("usage: ./irred_bridges_v1.py <bridge_num_file> <lower_bound_file>")
coeff = [-1]
mu = 0
for i in xrange(100):
  ouf1 = open(sys.argv[1], "a")
  ouf2 = open(sys.argv[2], "w")
  count = 0
  dfs([], 1, i)
  ouf1.write(str(i + 1) + " " + str(count) + "\n")
  coeff = [count] + coeff
  r = numpy.roots(coeff)
  r = r[numpy.isreal(r)]
  mu = 1/r[r>0][0].real
  ouf2.write(str(mu) + "\n")
  ouf1.close()
  ouf2.close()
```

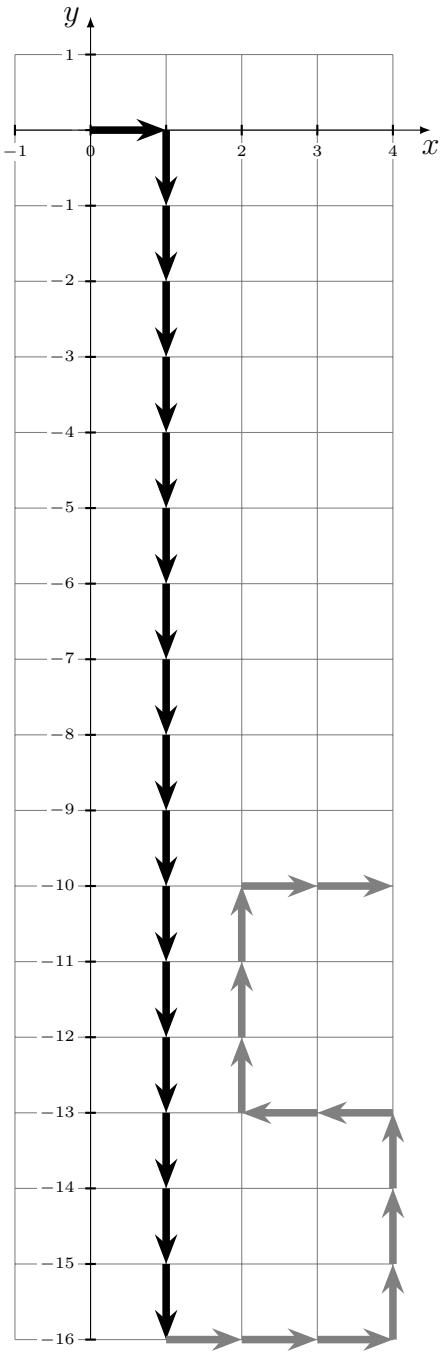

Figure 7: Prepend a 17-step walk to a 13-step irreducible bridge starting with a straight

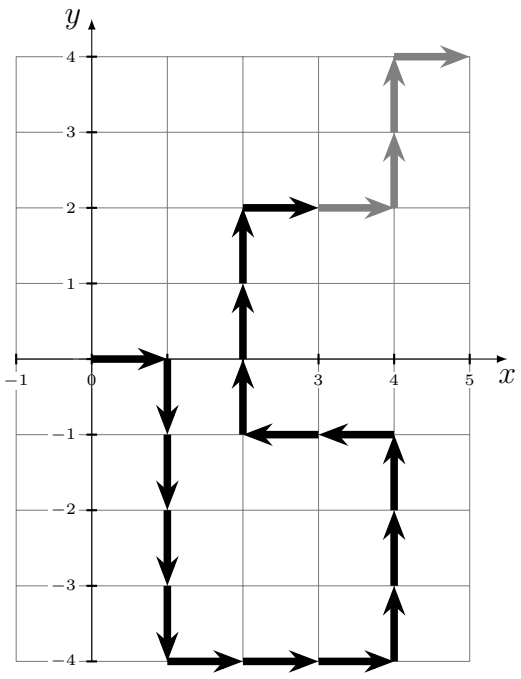

Figure 8: Prepend a 17-step walk to a 4-step irreducible bridge starting with a turn