Estimation of Connective Constant of Self-Avoiding Walks Using Monte Carlo Methods

Qu Qiancheng

March 2025

1 Introduction

A self-avoiding walk (SAW) on a two-dimensional square lattice is a sequence of moves on the lattice that does not revisit any previously occupied site. This concept is pivotal in statistical mechanics and polymer physics, as it models the behavior of linear polymers in a solvent. The primary challenge in studying SAWs lies in determining the number of distinct walks, denoted as c_L, for a given length L. This number grows exponentially with L, approximately as $c_L \approx \mu^L$, where μ is the connective constant. Accurately estimating μ is essential for understanding the statistical properties of SAWs.

1.1 Problem Statement

Consider the two-dimensional square lattice $\mathbb{Z}^2$. A self-avoiding walk (SAW) of length L is defined as a sequence of $L+1$ distinct vertices $(x_0, y_0), (x_1, y_1), \ldots, (x_L, y_L)$ in $\mathbb{Z}^2$, where each successive vertex is a nearest neighbor of the preceding one. Assuming the walk begins at the origin, i.e., $(x_0, y_0) = (0, 0)$, the objective is to determine the number of such distinct walks, denoted as c_L. Due to the combinatorial explosion of possible paths as L increases, exact computation of c_L becomes infeasible for large L. Therefore, developing efficient algorithms to estimate c_L and, consequently, the connective constant μ is of significant interest.

2 Basic Deterministic Methods

To establish a benchmark for our estimations, we employed a recursive algorithm to compute c_L for small values of L. This method systematically explores all possible self-avoiding walks of a given length. The results for $L = 1$ to 20 are presented in Table 1.

These exact counts serve as a foundation for validating the accuracy of Monte Carlo estimation methods applied to longer walks.

Table 1: Number of self-avoiding walks c_L for lengths $L = 1$ to 20.

Length L	Number of SAWs c_L
1	4
2	12
3	36
4	100
5	284
6	780
7	2,124
8	5,988
9	16,524
10	45,988
11	127,912
12	353,800
13	976,248
14	2,692,537
15	7,417,582
16	20,354,837
17	55,366,322
18	150,879,864
19	410,934,856
20	1,118,725,100

3 Naïve Monte Carlo Method

A direct approach is to generate random walks of length L uniformly from the set of all possible walks. Since a standard random walk on the square lattice has 4^L possibilities, the probability of generating a SAW is given by:

$$P_{\text{accept}} = \frac{c_L}{4^L}. \tag{1}$$

For large L, c_L grows approximately as μ^L with $\mu \approx 2.6$, so the acceptance probability becomes:

$$P_{\text{accept}} \approx \left(\frac{\mu}{4}\right)^L. \tag{2}$$

Since the acceptance rate decays exponentially with L, the naïve uniform sampling approach is highly inefficient.

3.1 Simulation Results and Analysis

With 100,000 trials and $L = 20$, our Monte Carlo estimation yielded:

$$\begin{aligned} \hat{c}_L &\approx 835,628,837.11, \\ \hat{\mu} &\approx 2.793191, \\ \text{SAW ratio} &\approx 0.000760. \end{aligned}$$

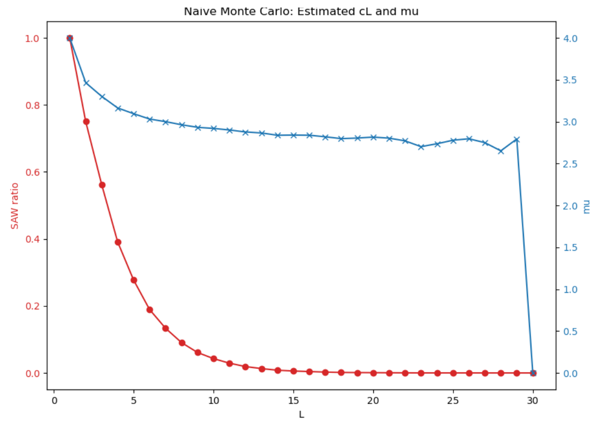

Figure 1: Naïve Monte Carlo simulation results for μ estimation.

However, for $L = 30$, the observed SAW ratio dropped to zero, leading to an estimated $\mu = 0$.

This issue arises due to the exponentially decreasing acceptance probability. Given that $P_{\text{accept}} \approx (\mu/4)^L$, for $L = 30$ we expect:

$$P_{\text{accept}} \approx \left(\frac{2.6}{4}\right)^{30} \approx 10^{-6}. \tag{3}$$

With a finite number of trials (e.g., 10^5), it is highly probable that no valid SAWs are sampled at all, causing $\hat{c}_L$ and $\hat{\mu}$ to be erroneously estimated as zero. This demonstrates the breakdown of the naïve Monte Carlo method at larger values of L and reinforces the necessity of using more efficient sampling techniques.

3.2 Variance of the Estimator

//TODO: we derive the variance

4 The Rosenbluth Algorithm

4.1 Importance Sampling and the Normalization Factor

Our objective is to estimate the normalization constant c_L. The unnormalized density is

$$p(z_{0:L}) = \mathbf{1}\{z_{0:L} \text{ is a SAW}\}. \tag{4}$$

Thus, the normalization factor is given by

$$c_L = \int p(z_{0:L})\, dz_{0:L}. \tag{5}$$

Using importance sampling method, if we sample from a biased proposal distribution $q(z_{0:L})$, we write

$$c_L = \int \frac{p(z_{0:L})}{q(z_{0:L})}\, q(z_{0:L})\, dz_{0:L} = \mathbb{E}_q\left[\frac{p(z_{0:L})}{q(z_{0:L})}\right]. \tag{6}$$

We define the importance weight as

$$W(z_{0:L}) = \frac{p(z_{0:L})}{q(z_{0:L})}. \tag{7}$$

An unbiased estimator for c_L is given by averaging weights over N samples:

$$\hat{c}_L = \frac{1}{N}\sum_{j=1}^{N} W(z_{0:L}^{(j)}). \tag{8}$$

4.2 Algorithm Description

Starting at the origin $z_0 = (0,0)$, the algorithm proceeds as follows:

1. **Initialization:** Set the initial weight $W = 1$ and $z_0 = (0,0)$. Let the visited set be $\mathcal{V} = \{z_0\}$.
2. **Growth Process (for each step $i = 0, 1, \ldots, L-1$):**
   - Determine all valid (unvisited) neighboring sites. Let m_i be their count.
   - If $m_i = 0$, set $W = 0$ and terminate the trial.
   - Otherwise, choose one neighbor uniformly at random and update z_{i+1}.
   - Update weight:
   $$W \leftarrow W \times m_i. \tag{9}$$
3. **Result:** After L steps, if $W > 0$, the final weight is

   $$W(z_{0:L}) = \prod_{i=0}^{L-1} m_i. \tag{10}$$

   The probability of generating this path using q is

   $$q(z_{0:L}) = \prod_{i=0}^{L-1} \frac{1}{m_i}. \tag{11}$$

   Since

   $$q(z_{0:L})W(z_{0:L}) = \prod_{i=0}^{L-1} \frac{1}{m_i} \times \prod_{i=0}^{L-1} m_i = 1, \tag{12}$$

   the weight W corrects for the bias introduced by sequential selection.

4.3 Simulation Result

With 100,000 trials and $L = 20$, our Monte Carlo estimation yielded:

$$\hat{c}_L \approx 893627935.66,$$
$$\hat{\mu} \approx 2.802579,$$
$$\text{sampling success ratio} \approx 0.911090.$$

//TODO: add and explore the graph: "Monte Carlo II (Rosenbluth): Estimated cL and mu"

4.4 Variance of the Estimator

//TODO: we derive the variance

4.5 Algorithmic Performance with Naive Monte Carlo

//TODO: we compare the estimation using Rosenbluth Algorithm with he actual c_L generated by deterministic methods, and also from the naive monte carlo. Then compare the computation time; we'll use several figures to present the result.

5 Metropolis-Hastings Algorithm

5.1 Application of Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is a Markov Chain Monte Carlo (MCMC) method used to sample from a target distribution by constructing a Markov chain whose stationary distribution is the desired distribution. The general procedure of the MH algorithm is summarized as follows:

1. Initialize time step $t = 1$.
2. Set an initial state $\theta^{(t)} = u$, where u is an appropriately chosen starting value.
3. Repeat the following steps:
   (a) Increment t: $t \leftarrow t + 1$.
   (b) Propose a candidate state θ^* sampled from a proposal distribution $q(\theta^* \mid \theta^{(t-1)})$.
   (c) Compute the acceptance probability:

$$\alpha = \min\left(1, \frac{p(\theta^*)}{p(\theta^{(t-1)})} \times \frac{q(\theta^{(t-1)} \mid \theta^*)}{q(\theta^* \mid \theta^{(t-1)})}\right). \tag{13}$$

   (d) Generate a uniform random variable $a \sim \text{Uniform}(0, 1)$.

(e) If $a \leq \alpha$, accept the proposal: $\theta^{(t)} = \theta^*$; otherwise, reject it: $\theta^{(t)} = \theta^{(t-1)}$.

4. Repeat until $t = T$.

For a SAW of length L on a 2-dimensional regular lattice with coordination number $z = 4$, we fix one end of the walk at the origin and denote the movable end as I. The Metropolis-Hastings algorithm is implemented as follows:

1. At each step, the proposal involves either growth or shrinkage with equal probability.

2. If the move is a growth move, one of the $z - 1$ non-occupied neighboring edges of I is randomly selected for expansion. The length increases to $L + 1$.

3. If the move is a shrinkage move, the occupied edge incident to I is converted into an unoccupied state. The length decreases to $L - 1$.

The acceptance probability (13) mentioned above for a growth move, denoted as P_+, is given by:

$$P_+ = \min\left(1, \frac{\frac{1}{C_{L+1}}}{\frac{1}{C_L}} \times \frac{\frac{1}{2}}{\frac{1}{2} \times \frac{1}{z-1}}\right), \tag{14}$$

where, for our specific case where $z = 4$, this simplifies to:

$$P_+ = \min\left(1, \frac{3}{\mu}\right). \tag{15}$$

Similarly, the acceptance probability for a shrinkage move, denoted as P_-, is given by:

$$P_- = \min\left(1, \frac{\frac{1}{C_{L-1}}}{\frac{1}{C_L}} \times \frac{\frac{1}{2} \times \frac{1}{z-1}}{\frac{1}{2}}\right), \tag{16}$$

which simplifies to:

$$P_- = \min\left(1, \frac{\mu}{3}\right). \tag{17}$$

Additionally, special attention is required for the case where the walk length is $L = 0$ (i.e., a null walk). In this case, there are z empty incident edges instead of $z - 1$. For simplicity, we adopt the convention proposed in [1], where an edge incident to the original site is permanently deleted.

//TODO: add the final stage, critical point, etc.

5.2 Estimating the Growth Constant μ

//TODO: add grand canonical estimation, i.e, the way to estimate μ using MH.

Appendix

For further details and access to the complete codebase, please refer to our project repository on GitHub:
https://github.com/vincent-wat04/SAW_growth_constant_estimation

References

[1] H. Hu, X. Chen, and Y. Deng, *Irreversible Markov chain Monte Carlo algorithm for self-avoiding walk*, Frontiers of Physics, vol. 12, p. 120503, 2016.