# Implementation of the Wang Landau Algorithm to Cubic Lattice Knots


Zihao Zhu

Department of Mathematics

University of California, Davis

*Supervisor*

Mariel Vazquez

In partial fulfillment of the requirements for the degree of

*Bachelor of Science in Mathematics*

September 13, 2019

# Abstract

The mathematical study of circular polymers in biology and medical science often involves modeling them as 3-dimensional self avoiding polygons as using sampling algorithm to understand their behaviour. The BFACF Algorithm is an ergodic Markov Chain Monte Carlo method that samples the space of self-avoiding lattice polygons of fixed knot type and varying length. However, using this method to generate large independent and identically distributed(iid) sets of polygons is expensive computationally and challenging as the length of the chain increase. With the objective to improve the sampling efficiency and expand the functionality of the sampling process, we implement the Wang Landau algorithm. This algorithm provides an improved way to sample polygons related by BFACF moves; also, it samples uniformly with respect to different energies, such as writhe and bending energy. In this work, we compared the sampling distribution obtained by each algorithm for a variety of knot types and lengths. The comparison shows high coherency between these two sampling methods, which validates the implementation. The Wang Landau algorithm provides a more efficient and precise tool that helps us understand the behavior of polymers. Sampling data obtain by Wang Landau algorithm shows the relative probability of pairwise energy states and overall distribution for all energy states.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Computer-based simulations for the modeling of polymers in morbid science or of biopolymers such as DNA have gained increased attention in the past decades [4]. With improved computation power, scientists have the ability to do virtual experiments. In the field of Microbiology and Molecular Genetics, scientists have been trying to understand the underlying biological process of disease. Cancer studies, as one of the main focus, involve large scale computer simulations to reveal biological information from experimental data. For example, Lai et al(October 2007) introduced a algorithm analyzing array comparative genome hybridization data (aCGH), which successfully explored the region of aberration and applied to breast cancer studies[5]. The computer simulations is the key tool to extract information, provide complex analysis and support hypotheses

Computer simulations rely on mathematical models. For example, if we are interested in a low resolution study of the topology of a DNA molecule, then the DNA double helix can be modeled as the curve drawn by the axis of the helix. If the molecule is circular the resulting curve is an embedding of a circle in 3-dimensional space. In mathematics, this is the definition of a knot . The

model can be discretized to represent DNA as a circular polygon, which could be knotted. This is the main model for this thesis.

The BFACF algorithm (Berg and Foester 1981[1], Aragao de Carvalho et al 1983[6], Aragao de Carvalho and Caracciolo 1983[7]) was proposed in 1991 to operate on polygons doing stochastic sampling[8]. It has been used for modeling circular DNA as self-avoiding polygons in the simple cubic lattice ($\mathbb{Z}^3$)[8]. This algorithm was developing through multiple stages from generating statistical ensembles of paths (or surfaces) with a Boltzmann probabilistic weight[1], applying to bosonic walks and fermionic walks[1] and then applying to self-avoiding walk [6][7]. The fundamental definitions, such as the elementary moves of the BFACF algorithm, the Markov properties and the limiting distribution are described in section 2.3. The length of the polygons (the number of edges) is the dominant factor in the limiting distribution, and therefore is an important component of the sampling results. Scientists explored more perspectives to inspect the behaviour of DNA, so that length driven BFACF algorithm restricts other angles to analyze modeled DNA. Two major deficiencies are that the BFACF algorithm cannot generate sufficient samples in low probability area and the length driven BFACF algorithm cannot utilize attributes other than length to guide the sampling.

BFACF is a Markov Chain Monte Carlo algorithm (MCMC), which is a stochastic process that approximates the distribution of memoryless sequential random events applied to polygons. The stochastic process is similar to placing a person in a strange town and the person will walk randomly with pre-defined probabilities of each direction in every intersection. An example method that defines probabilities could be flipping a coin in the intersection to decide which direction to go. For the modeled DNA polygon, each specific configuration of a polygon is a conformation that contains analyzable information, length of the polygon for example. An example of analyzable information for a random walk

in the city could be the distance the person has walked. Thus, it is possible that different conformations share the same properties, for example two different polygons can have the same length. The BFACF algorithm allows for uniform sampling of the space of SAPs of fixed length and knot type. BFACF genrates samples by performing three possible moves. In BFACF, those moves are +2, -2, +0 moves, and they are applied following probabilities pre-determined by the model.

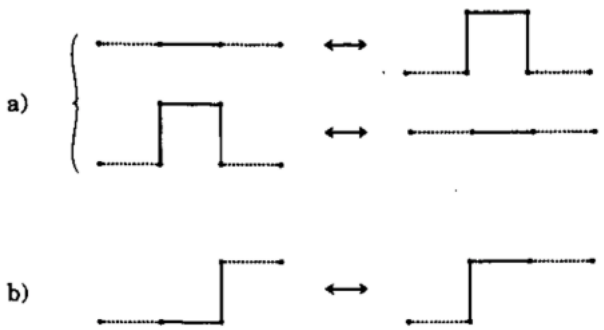



Figure 1.1: The elementary moves of the BFACF algorithm [1]:(+2),(-2),(+0)

In the case of a random walk in a strange city, one could ask how probable it is for the person to stay at a mall. With BFACF, after fixing a knot type, the algorithm can approximate the average length, the probability that the knot is at a specific length, as well as other attributes(writhe, bending energy). Moreover, since the algorithm provides samples from a wide range of lengths, we can pick target samples to perform further operations.

While the BFACF provides wide length range samples, most of the samples are concentrated around the average length. This results in inefficient sampling from the trails of the distribution (i.e. chains of lengths far removed from the mean). This deficiency is caused by the definition of the distribution, which is dominated by length value of the polygonal knots. Consequently, the BFACF

algorithm is strongly related to the length of the polygons. During the sampling process, the conformation either maintains the length, decreases the length by 2, or increases the length by 2. After the user chooses a parameter that defines the distribution of +2, -2, +0 move types, the stochastic process will depend on this distribution. Hence, with sufficiently many samples, the distribution of lengths follows a bell shape curve with the majority of the samples concentrated around the average length. The samples whose length deviates from average length are infrequent, so that we have to rerun the simulation with different parameters, adjusting the average length close to the target length, or increase the total sampling size for additional samples deviating from the average length. Any of these choices requires notable larger computation.

Another limitation of the BFACF algorithm is that it is tied to the length of the polygon and in its classical form it cannot sample with respect to other properties meaning it cannot use writhe or other energy to decide the probability of move types. When modeled DNA, there are abundant significant properties other than length(i.e. energy and use energy in future reference), such as bending energy, writhe, etc. Depending on the application, users may want to bias their sampling by limiting bending energy, or generate a uniform samples of polygons with a specific average writhe. Deriving samples with respect to other energies is critical for particular research. For instance, when studing DNA packaging in the bacteriophage capsids, Arsuaga et al.(2005) found that that the writhe of DNA inside is negatively biased. The ability to generate polygon with negative writhe is relevant to the 2005 DNAs paper(Arsuaga et al.).

In this work, we utilize another Markov Chain algorithm, Wang-Landau algorithm to improve the sampling process. The Wang-Landau algorithm is a general Markov Chain algorithm that can be applied to our specific context of modeling DNA as polygons[9][10]. Instead of using the distribution from the BFACF algo-

rithm, the Wang-Landau algorithm defines a uniform distribution, which means different energy states are distributed most evenly in all the samples. In the Wang-Landau stochastic process, the possible move types are still +2, -2, +0 moves defined by the BFACF algorithm, but the Wang-Landau algorithm determines which move to take. This decision depends on the density estimation along the sampling. In the BFACF algorithm, the distribution of move types is defined first, a random number is generated and probabilities determined by the underlying distribution are used to perform one of the move types. Instead, Wang-Landau utilizes a Metropolis Hastings algorithm, a decision method where a random move type is chosen before the algorithm decides whether to accept the decision based on acceptance probabilities.

In the actual implementation, the Wang-Landau algorithm breaks into two phases - training and sampling. The training phase of the Wang Landau algorithm takes a random walk, keeps exploring the sample space and records approximate density along the walk. Then in the sampling phase, it utilizes the trained result from the training phase to generate target samples. Thus, the algorithm introduces two key components - the weight list and observation counter. The ratio of weights approximate the ratio of energy densities in the sample space and the observation counter tells the number of times the process observes particular energy. The weight list in training phase helps the decision whether to accept a move type and the observation provides confidence in how well the process understands the sample space. Still taking the example that puts a person in a strange city, the person starts with a random place in the city with a notebook writing down how many times he has been to a place. Every time the person goes to another place, he checks if he has been here before and either writes down a new place name or adds one more observation time. in order to get familiar with the strange city faster, the person wants to explore new places that he has never been

to. Thus, after the person chooses a direction to go, he will check the notebook to see whether he has been to the next place many times. in order to reduce the times he has been to a visited place, the person will reject the move that takes him to a familiar place. Similar to the Wang Landau algorithm, it is trained to understand the sample space so that next time the person comes back to the city with the notebook he knows how to get to a region of the city. Also, the person can choose what properties to record, such as the category of the place if it is a restaurant or shopping mall, size of the space, color of the space, etc.

The distinctive sampling mechanism defined by the Wang Landau algorithm explores the whole sample space more evenly and it allows a diversity of energy perspectives. Thus, the Wang Landau algorithm solves an issue that BFACF cannot handle. Consequently, in the context of modeling polymers, the Wang Landau algorithm is an improved sampling technique.

# Chapter 2

# Background

## 2.1 Knot theory

In this work, the Wang-Landau algorithm is applied to lattice knots. Lattice knots are self-avoiding polygons SAPs of finite length such that each vertex is in $\mathbb{Z}^3$, and all edges are of length 1 and parallel to one of the coordinates axes.

**Definition 2.1.1** ([11])**.** A link $L$ of $m$ components is a subset of $\mathbb{S}^3$, or of $\mathbb{R}^3$, that consists of $m$ disjoint, piece-wise linear, simple closed curves. A link of one component is a knot.

**Definition 2.1.2** ([8])**.** An embedding is a map $f : X \to Y$ In this case $f$ is one-to one and structure-preserving.

Let $\mathbb{S}^1$ be the unit circle. A knot is also defined as an embedding $f : \mathbb{S}^1 \to \mathbb{R}^3$. Double stranded circular DNA can be represented as a knot by tracing the center axis of the DNA; also, it can be represented as a link regarding each strand is a knot.

**Definition 2.1.3** ([11])**.** Links $L_1$ and $L_2$ in $\mathbf{S}^3$ are equivalent if there is an orientation-preserving piecewise linear homeomorphism $h$: $\mathbf{S}^3 \to \mathbf{S}^3$ such that

$h(L_1) = (L_2)$

We call the equivalent classes the *knot types*.

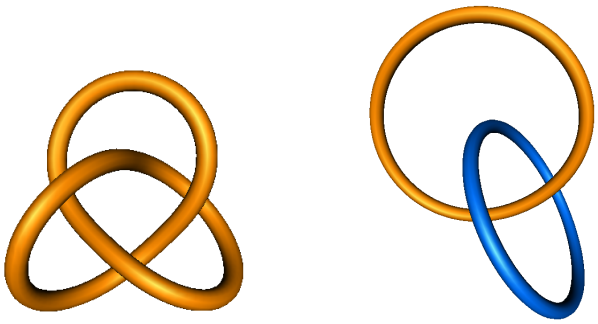


Figure 2.1: Trefoil knot $3_1$ and 2-component link $2_1^2$ (Hopf link)

There are three significant quantities associated to the topology and geometry of an embedding of a link - Linking number, Twist and Writhe. They are closely related to each other. Linking number is a topological invariant(given orientation), which means that any two equivalent links share the same linking number. Twist and Writhe however are not topological invariants, which means the value of Twist and Writhe may change from one embedding to another even when the link type is preserved. The important result is that the sum of twist and writhe equals the linking number:

**Theorem 2.1.1** (White's Theorem[12])**.** *Given a link L,*

$$Lk(L) = Tw(L) + Wr(L) \tag{2.1}$$

This result suggests that since linking number is a topological invariant and therefore is preserved under deformations of the link, adding twist to the knot will introduce writhe changes[12].

### 2.1.1 Linking Number

Linking number measures the extent that components of a link interwine with each other[13]. When computing $Lk$, the crossings are mapped to numerical values. Depending on the orientation of strands on a crossing, the crossings can be classified into two classes and each class we will assign a +1 or a -1 value(2.2).

(a) +1 (b) -1

Figure 2.2: Two types of crossings

There are two methods for calculating the linking number mainly: Gauss integral or via projection.

Since the linking number is a topological invariant, it is more easier to compute the linking number by choosing a projection of the link. After chosing a projection and orientations for each component, to obtain the linking number sum all the +1 and -1, and divide the total by 2.

$Lk$ can be seen as a summation of crossings over two components, and expressed as a Gauss integral as in the next definition.

**Definition 2.1.4.** the linking number of a link $L$ can be obtained as follows:

$$Lk(L)\mathbf{Lk}(c_1, c_2) = \frac{1}{4\pi} \oint_{c_1} \oint_{c_2} \frac{\mathbf{r}_2 - \mathbf{r}_1}{|\mathbf{r}_2 - \mathbf{r}_1|^3} \cdot (d\mathbf{r}_1 \times d\mathbf{r}_2) \tag{2.2}$$

$c_1, c_2$ are the two component of the link and $\mathbf{r}_1, \mathbf{r}_2$ are points along components of the link. The integral is the summation of the solid angle.

**Definition 2.1.5.** The solid angle is defined as:

$$d\mathbf{\Omega}^*(\mathbf{r}_1, \mathbf{r}_2) \equiv d\mathbf{\Omega}^*_{12} = \frac{|d\mathbf{r}_1^{\perp}| \cdot |d\mathbf{r}_2^{\perp}| \cdot sin\alpha}{|\mathbf{r}_2 - \mathbf{r}_1|^2} \tag{2.3}$$

$|d\mathbf{r}_1^{\perp}|$ and $|d\mathbf{r}_2^{\perp}|$ are the perpendicular components of $\mathbf{r}_1 - \mathbf{r}_2$, the $|\cdot|$ is the Euclidean metric, $\alpha$ is the angle between $|d\mathbf{r}_1^{\perp}|$ and $|d\mathbf{r}_2^{\perp}|$. Thus $|d\mathbf{r}_1^{\perp}| \cdot |d\mathbf{r}_2^{\perp}| \cdot sin\alpha$ is indicating the parallelogram area formed by $|d\mathbf{r}_1^{\perp}|$ and $|d\mathbf{r}_2^{\perp}|$.

Since the solid angle is always positive, we can connect the signed solid angle to the solid angle by the +1 and -1 value assigned to the crossings. Let the $w_{12}$ be the +1 or -1 value assigned to crossings then:

**Definition 2.1.6.** The signed solid angle is given by

$$d\mathbf{\Omega}_{12} = w_{12} \cdot d\mathbf{\Omega}^*_{12} = \frac{(d\mathbf{r}_1 \times d\mathbf{r}_2) \cdot (\mathbf{r}_2 - \mathbf{r}_1)}{|\mathbf{r}_2 - \mathbf{r}_1|^3} \tag{2.4}$$

We can easily show that the signed solid angle is the absolute value of the solid angle:

$$\begin{aligned} \frac{|d\mathbf{r}_1^{\perp}| \cdot |d\mathbf{r}_2^{\perp}| \cdot sin\alpha}{|\mathbf{r}_2 - \mathbf{r}_1|^2} &= \frac{|d\mathbf{r}_1^{\perp}| \cdot |d\mathbf{r}_2^{\perp}| \cdot sin\alpha \cdot 1}{|\mathbf{r}_2 - \mathbf{r}_1|^2} \\ &= \frac{||d\mathbf{r}_1^{\perp}| \cdot |d\mathbf{r}_2^{\perp}| \cdot sin\alpha \cdot \mathbf{u} \cdot (\mathbf{r}_2 - \mathbf{r}_1)|)}{|\mathbf{r}_2 - \mathbf{r}_1|^2 \cdot |\mathbf{r}_2 - \mathbf{r}_1|} \\ &= \frac{|(d\mathbf{r}_1 \times d\mathbf{r}_2) \cdot (\mathbf{r}_2 - \mathbf{r}_1)|}{|\mathbf{r}_2 - \mathbf{r}_1|^3} \end{aligned} \tag{2.5}$$

The full solid angle is $4\pi sr$, so that if we divide the signed solid angle by the full solid angle $4\pi sr$, it represents the half of the average contribution of the evaluated point $\mathbf{r}_1, \mathbf{r}_2$ over all possible directions. Thus the double line integral is the summation of all points along the components at all possible directions.

Then we can express the linking number Gauss integral as[14] :

$$\mathbf{Lk}(c_1, c_2) = \frac{1}{4\pi} \oint_{c_1} \oint_{c_2} d\mathbf{\Omega}_{12} \tag{2.6}$$

### 2.1.2 Twist and Writhe

Twist and writhe are measurement related to coiling between two components of a link. The ideas of twist and writhe are more intuitive when applying to double stranded DNA. Twist measures how one strand of the double stranded DNA coils around the another strand. Writhe measures how the central axis of the double stranded DNA coils around itself. Therefore, the calculation of twist is the integral of the rotation rate along the center axis. For the purpose describing the integral, we parametetrize the closed curve by $t$: $\mathbf{r}(t)$ is the central axis of the closed curve; $\mathbf{a}(t)$ is the unit vector that perpendicular to the central axis; $\mathbf{s}(t)$ is the closed curve; $\epsilon$ is the distance from the curve to central axis.

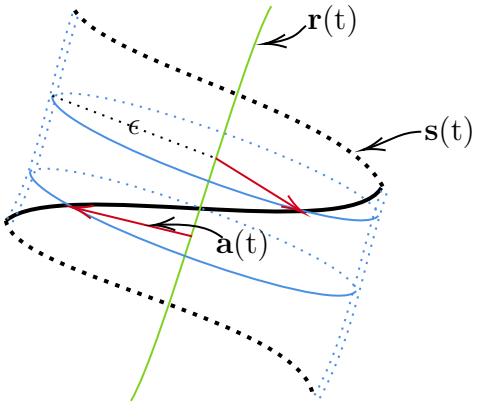



Figure 2.3: Interpretation of the Twist

The Twist measures how much the black curve winds about the green curve

**Definition 2.1.7.** The angle of rotation is defined as:

$$d\phi = (\frac{d\mathbf{r}}{dt} \times \mathbf{a}) d\mathbf{a} \tag{2.7}$$

$(\frac{d\mathbf{r}}{dt} \times \mathbf{a})$ defines the positive unit vector so that $d\phi$ is the small angle that rotates $\mathbf{a}(t)$.

The integral over the parameter $t$ along the whole length of the closed curve is the Gauss integral of the Twist:

**Definition 2.1.8.** Given a two component link $L$

$$\mathbf{Tw}(L) = \frac{1}{2\pi} \int_{t=0}^{L} (\frac{d\mathbf{r}}{dt} \times \mathbf{a}) \frac{d\mathbf{a}}{dt} dt \tag{2.8}$$

For Writhe calculation, given a link we either consider the self-writhe of each component, or, in the case of DNA, we compute the writhe of the curve defined by the axis of the helix. The writhe is the average of the projected writhe over all projection direction. To compute the projected writhe we consider a diagram of the curve, sum up the +1, -1 and divide the total by two.

**Definition 2.1.9.** The writhe of a link can be expressed as a Gauss integral:

$$\mathbf{Wr}(c_1, c_2) = \frac{1}{4\pi} \oint_c \oint_c d\mathbf{\Omega}_{12} \tag{2.9}$$

Writhe is the half summation of the crossing value (+1,-1), note that the double line integral is counting the value twice, so that the coefficient remains $1/4\pi$.

## 2.2 Monte Carlo Method

A Monte Carlo method is an approximation of a deterministic distribution by large scale samples, which is widely used in physics and mathematics. For example, if there is a particular coin without fair assumption, instead of measuring the coin physically to see if it is fair, one can flip the coin in a large number of

times and keep counting the number of heads and tails. In the large number of trials of flipping the coins, the ratio between the number of heads and tails can approximate the fairness of the game.

In the eighteenth century, the model "Buffon's needle" by a French scientist Georges-Louis Leclerc, Comte de Buffon is one of the primary examples of the Monte Carlo method. The Buffon's needle consistes of dropping needles to parallel strips with same width, and estimating the probability of the needle across two strips.

There is a simplified example by using the Monte Carlo method

**Example 2.2.1.** Rolling Two Dices: Calculate the distribution of the sum of the value of rolled two dices. There are $6 \times 6$ possible outcomes from 2 to 12.

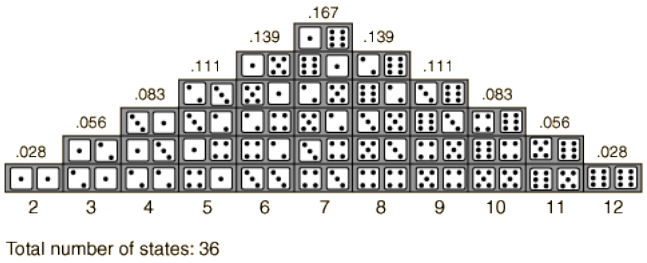



Figure 2.4: Sum distribution of rolling two dice[2]

number below the dice are the sum of two dice and the number above the dice is the associate probability

There are 36 outcomes in total and there is only one outcome where the sum is 2, so the probability of that the sum of rolled dice equals to 2 is $\frac{1}{36}$. Similarly for other probabilities of sum value. Instead of calculating the probabilities exactly, the Monte Carlo method will roll two dices in a large number of times. For example, if sum value equals to 3 rolled 5 times in 100 trails, the probability is approximated to 5%.

Figure 2.5 shows the results of a simulation of rolling dice 10,000 times

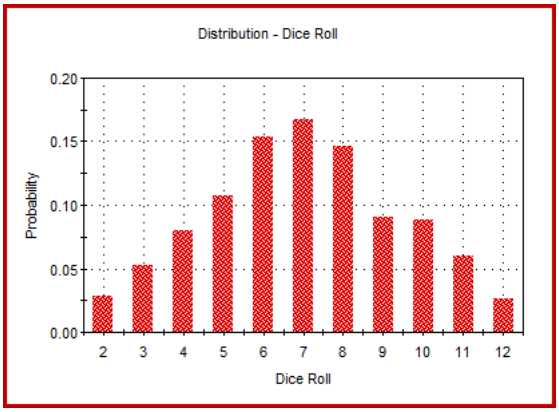



Figure 2.5: Monte Carlo simulation output for the sum of rolling two dice[3]

## 2.3 Markov Chains

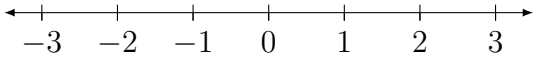
In this Chapter we will introduce Markov Chains and I will use *Lecture Notes for Introductory Probability*[15] Markov Chain is a stochastic process that describes a sequence of possible events. In general, the Markov Chain contains

- a set $S$ contains countable states, which are labels of particular states.
- transition probabilities that defines the probability from a state $i \in S$ to another state $j \in S$.
- the initial distribution of states.

With Markov Chain, the set of states can be either infinite or finite(since we only require $S$ to have countable states.

**Example 2.3.1.** A random walk in Z defines the set of states in $\mathbb{Z}$. In every particular state $i$, the probability of moving from $i$ to $i-1$ is $p$, and the probability of moving from $i$ to $i+1$ is $1-p$, $p \in [0,1]$. The initial distribution $\alpha$ is given by:

$$\mathbb{P}(X_0 = i) = \alpha_i = \begin{cases} 1, & \text{if } i = 0 \\ 0, & \text{otherwise} \end{cases}$$




**Example 2.3.2.** Taxi migration

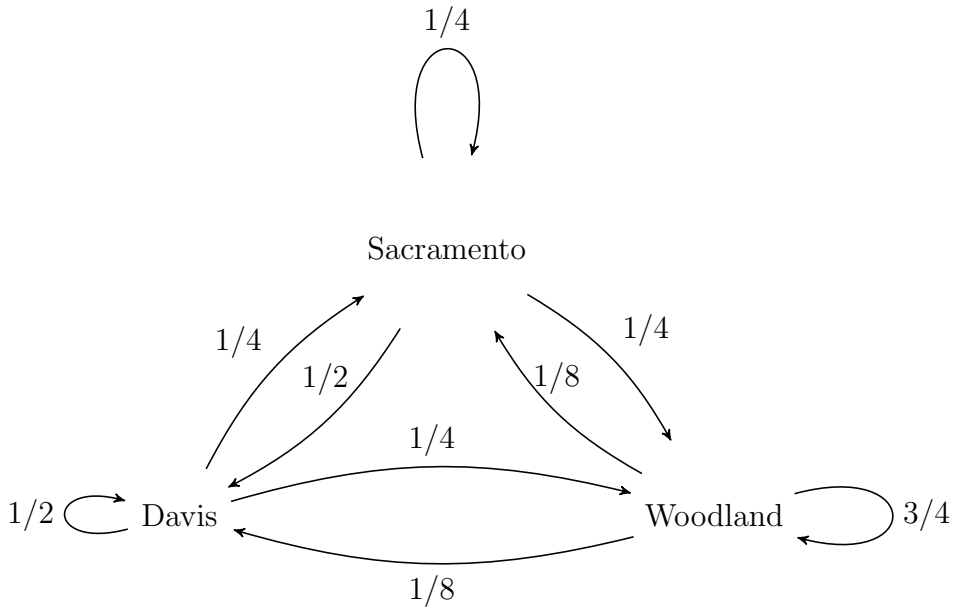



Figure 2.6: Taxi Markov Chain

This is an example of taxi migration in Davis, Woodland and Sacramento. Each arrow represent the direction along with the probability that a taxi moves between cities.

In this model, the states are 3 cities in California: Davis, Woodland and Sacramento. A directed arc indicates the transition probability that the taxi moves from a particular city to another city. With an initial distribution of the allocation of the taxis, the Markov Chain can mimic the migration over time.

All transition probabilities from state $i \in S$ to state $j \in S$ can be represented as a *transition matrix*.

represent transition probability in Matrix

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots \\ p_{21} & \ddots & \\ \vdots & & p_{nn} \end{bmatrix}$$

Transition probability that state $i$ to $j$ is a conditional probability that the probability to be at state $j$ conditioned on previous state is $i$.

$$P_{ij} = P(X_{n+1} = j | X_n = i)$$

By the law of total probability, the sum of the probabilities that a state transit to other states is one.

$$\forall i, \sum_{j=1}^{n} p_{ij} = 1$$

A Markov Chain has the property that the future and previous are independent, which means the future only depends on present state:

$$\mathbb{P}(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, Xn-1 = i_{n-2}, \ldots, X_0 = i_0) = \mathbb{P}(X_{n+1} = j | X_n = i)$$

**Definition 2.3.1.** Let $P_{ij}^n$ be the probability to start from i and end at j in n-steps

**Theorem 2.3.1.** *$P_{ij}^n$ is the $(i, j)$ entry of the nth power of the transition matrix, and represents the probability of reaching state $j$ often n steps starting at state $i$.*

*Proof.* Decompose the transition probability $P_{ij}^{n+m}$

$$\begin{aligned} P_{ij}^{n+m} &= P(X_{n+m} = j | X_0 = i) \\ &= \sum_k P(X_{x+m} = j, X_n = k | X_0 = i) \\ &= \sum_k P(X_{n+m} = j | X_n = k) \cdot P(X_n = k | X_0 = i) \\ &= \sum_k P(X_m = j | X_0 = k) \cdot P(X_n = k | X_0 = i) \\ &= \sum_k P_{kj}^n P_{ik}^m \end{aligned} \tag{2.10}$$

□

Thus the transition matrix $P^{m+n} = P^m P^n \Rightarrow P^n = P^1 P^1 \dots P^1 \Rightarrow$ the power of the transition matrix. If the initial distribution is $\alpha$ then the distribution at time n is $\alpha \cdot P^n$.

In the stochastic process, if the Masrkov Chain will converge to a steady state such that the next state will be the same state. In terms of linear algebra, the invariance distribution is the eigen vector correspongding the eigen value is one. Let $U$ be the Steady state, then by definition $U \cdot P = U$, thus $U(P - I) = 0$. Therefore linear algebra can be applied to find the steady state.

To see if the Markov Chain will converge, there are three properties to check: irreducible, aperiodic, and positive recurrent.

If state $i$ and state $j$ has positive transition probability from i to j also j to i, then state $i$ and $j$ are commutative and belong to the same *class*.

**Definition 2.3.2** (Irreducible)**.** The chain is irreducible if there is only one class.

The concept of aperiodic chain comes from period of the Markov chain, where period is a class property. If state $i$ has the period $d$ then the class that state $i$ belongs to has the period $d$

**Definition 2.3.3.** A state $i$ has *period* $d$ if $P_{ii}^n \geq 0$ implies $d|n$ and $d$ is the largest positive integer.

**Definition 2.3.4** (Aperiodic)**.** A state $i$ is aperiodic if state $i$ has *period* 1. The irreducible chain is aperiodic if all states have *period* 1.

For introducing positive recurrent we need to define *Return time* along with the probability mass function.

**Definition 2.3.5.** The Return time is defined as: $R_i = \inf\{n > 0, X_n = i\}$ where start with state $i$, $R_i$ is the first time return to the state $i$.

**Definition 2.3.6.** Then we can write out the probability mass function of the return time (p.m.f)

$$f_i^{(n)} = P(R_i = n | X_0 = i)$$

Thus, we can calculate the expectation of the time that starting with state $i$ and return to state $i$:

**Definition 2.3.7.** The expectation of the return time can be calculated as a sum as:

$$m_i = \mathbb{E}[R_i | X_0 = i] = \sum_{n=1}^{\infty} n f_i^{(n)}$$

**Definition 2.3.8.** State $i$ is positive recurrent if the expectation of the return time $m_i$ is finite. For an irreducible chain, we say that the chain is irreducible if all states are positive recurrent.

We call a chain *Ergodic* when the chain is Aperiod and Positive Recurrent.

**Theorem 2.3.2** (Convergence to invariant distribution)**.** *If a Markov chain is irreducible, aperiodic, and positive recurrent, then, for every i,j∈ S,*

$$\lim_{n \to \infty} P_{ij}^n = \pi_j$$

*, $\pi_j$ is the invariant distribution of state $j$*

**Theorem 2.3.3.** *Reversibility or Detailed Balance defined as: A Markov Chain with invariant distribution $\pi$ is reversible if and only if*

$$\pi_i P_{ij} = \pi_j P_{ji}, \forall i, j \in \Omega$$

Both Theorem 2.3.2 and 2.3.3 are necessary to ensure the desired distribution successfully constructed a Markov Chain. 2.3.2 implies there is an invariant distribution and if the desired distribution satisfies 2.3.3 then the desired distribution is invariant. We will check the BFACF distribution in the next section.

## 2.4 The BFACF Algorithm

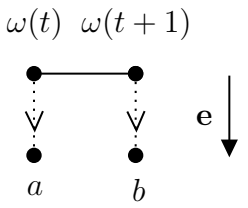
For the purpose of introducing the basics of the BFACF Algorithm, I will use *The Self-Avoiding Walk*[16]. BFACF is a Monte Carlo Markov Chain algorithm that operates on self-avoiding polygons (SAPs). In this work, we focus on SAPs in $\mathbb{Z}^3$. The self-avoiding polygons are corresponded to self-avioding walks.

**Definition 2.4.1** (N-step self-avoiding walk)**.** An N-step self-avoiding walk $\omega$ on $\mathbb{Z}^d$, beginning at the site $x$, is defined as a sequence of sites ($\omega(0)$,$\omega(1)$, ...,$\omega(N)$) with $\omega(0) = x$, satisfying $|\omega(j+1) - w(j)| = 1$, and $w(i) \neq w(j)$ for all $i \neq j$.

**Definition 2.4.2** (N-step self-avoiding polygon)**.** Let $N$ be an integer greater than 2. An N-step self-avoiding polygon is a set **P** of $N$ nearest-neighbour bonds with the following property: there exists a corresponding ($N$ -1)-step self-avoiding walk $\omega$ having $|\omega(N-1) - \omega(0)| = 1$ such that **P** consists of precisely the bond joining $\omega$( N - 1) to $\omega(0)$, and the $N$ - 1 bonds joining $\omega$( i - 1) to $\omega$( i) (i=1, ... ,$N$-1).

Janse van Rensbug and Whittington have proved that the aperiodic and positive recurrent class is the *knot types*[8] and that within a *knot type* the class is irreducible. Therefore, the BFACF algorithm applied to polygons satisfies the 2.3.2. The next step is to verify reversibility with desired distribution.

Before introducing the BFACF distribution, the formal definition of the elementary moves is needed. Let the current walk be $\omega$, choose one of the bonds $\omega(t)$ to $\omega(t+1)$, such that $\mathbf{e}$ is the unit vector perpendicular to the bond in one of $2d-2$ directions. Let $a, b$ be the lattice point $\omega(t)+\mathbf{e}, \omega(t+1)+\mathbf{e}$.




The options split into 3 cases:

1. $a \neq \omega(t-1)$ and $b \neq \omega(t+2)$. The new walk adds two bonds and the new walk is $\widetilde{\omega} = (\dots, \omega(t), a, b, \omega(t+1), \dots)$.

2. $a = \omega(t-1)$ and $b = \omega(t+2)$. The new walk reduces two bonds and the new walk is $\widetilde{\omega} = (\dots, \omega(t-1), \omega(t+1), \dots)$.

3. $a \neq \omega(t-1)$ and $b = \omega(t+2)$. The new walk keeps same bond number and the new walk is $\widetilde{\omega} = (\dots, \omega(t), a, \omega(t+2)\dots)$. Similarly, $a = \omega(t-1)$ and $b \neq \omega(t+2)$ then the new walk is $\widetilde{\omega} = (\dots, \omega(t-1), b, \omega(t+1)\dots)$.

**Definition 2.4.3.** BFACF distribution is defined as:[1]

$$\pi(\sigma) = \frac{|\sigma| \cdot z^{|\sigma|}}{\sum_{i=1}^{\infty} i \cdot z^i \cdot \mu_i(k)} \tag{2.11}$$

$\pi(\sigma)$ is the invariant distribution of the conformation $\sigma$, $z$ is the "fugacity" parameter that adjusts the distribution and probability, $\mu_i(k)$ is the number of

length $i$ conformations with knot type $k$. Thus, $\sum_{i=1}^{\infty} i \cdot z^i \cdot \mu_i(k)$ is the normalization.

Construct the Markov chain by satisfying the detailed balance condition 2.3.3.

Plug in detailed balance: suppose $|\sigma| + 2 = |\sigma'|$ WLOG

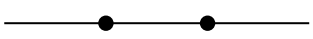
$$\frac{|\sigma| \cdot z^{|\sigma|}}{\sum_{i=1}^{\infty} i \cdot z^i \cdot \mu_i(k)} \cdot \frac{1}{|\sigma|} \cdot p(+2) = \frac{(|\sigma| + 2) \cdot z^{|\sigma|+2}}{\sum_{i=1}^{\infty} i \cdot z^i \cdot \mu_i(k)} \cdot \frac{1}{|\sigma| + 2} \cdot p(-2) \tag{2.12}$$

The transition probability from $\sigma \to \sigma'$ is decomposed by choosing one of the bonds, choosing one of the directions and one of the elementary moves. Equation 2.12 could be simplified to

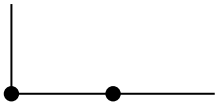
$$p(+2) = z^2 \cdot p(-2) \tag{2.13}$$

Similarly, if $|\sigma| = |\sigma'|$, The right hand side canceled with the left hand side so that detailed balance left $p(+0) = p(+0)$. Thus, the constraint for $p(+0)$ is the total probability that the sum of the probabilities of all possible moves has to be less or equal to 1. There are 4 cases for 3d self-avoiding polygons to check if the sum is less or equal to 1. Let $E$ be the set of all bonds such that $E(t)$ is the segment from $\omega(t)$ to $\omega(t+1)$, $t \in [0, |\omega| - 1]$. let $E(t)$ be chosen bond to perform elementary moves:

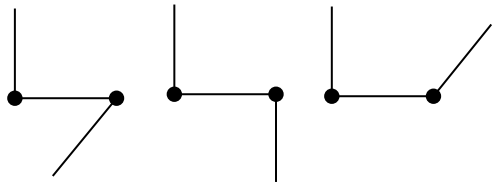
1. $E(t-1)$ and $E(t+1)$ are parallel to $E(t)$



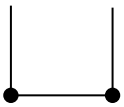
2. one of the $E(t-1)$ and $E(t+1)$ is perpendicular to $E(t)$



3. both $E(t-1)$ and $E(t+1)$ are perpendicular to $E(t)$ and $E(t-1)$ is not *anti-parallel*(parallel and the directions of two bonds are opposite) to $E(t+1)$ given orientation



4. both $E(t-1)$ and $E(t+1)$ are perpendicular to $E(t)$ and $E(t-1)$ is anti-parallel to $E(t+1)$ given orientation



In case 1, all possible moves are +2 moves $\Rightarrow 4p(+2) \leq 1$. In case 2, one direction is +0 move and all other moves are +2 moves $\Rightarrow p(+0) + 3p(+2) \leq 1$. In case 3, two directions correspond to +0 moves and the other two directions correspond to +2 moves $\Rightarrow 2p(+0) + 2p(+2) \leq 1$. In the case 4, one directions corresponds to a -2 move and all other moves correspond to +2 moves $\Rightarrow p(-2) + 3p(+2) \leq 1$. The intersection of inequalities forms the *standard choice*:

$$\begin{aligned} p(+2) &= \frac{z^2}{1+(2d-3)z^2} \\ p(-2) &= \frac{1}{1+(2d-3)z^2} \\ p(+0) &= \frac{1+z^2}{2[1+(2d-3)z^2]} \end{aligned} \tag{2.14}$$

For convergence, the 2.11 has to be finite.
Apply the root test to $\sum_{i=1}^{\infty} i \cdot z^i \cdot \mu_i(k)$:

$$\lim_{n\to\infty} (n \cdot z^n \cdot \mu_n(k))^{\frac{1}{n}} = z \cdot \lim_{n\to\infty} \mu_n(k)^{\frac{1}{n}} = z \cdot \mu$$

if $z \cdot \mu < 1$ then the series converge $\Rightarrow z \leq \frac{1}{\mu}$. We call $\mu$ the critical value.

*BFACF Algorithm* The algorithm generates $\omega^{(t)}$, where $\omega^{(t)}$ is the conformation generated at time $t$:

1. Start with initial state $\omega^{(0)}$
2. Choose a bond $e$ uniformly from $\omega^{(0)}$
3. Consider one of 2d-2 (d is the dimension of the walk) directions perpendicular to the bond $e$, and choose one of the elementary moves with the probabilities $p(+2), p(-2), p(+0)$
4. Check the self-avoiding property. If the chosen elementary move is not against self-avoidance, then perform the move. Otherwise, $\omega^{(t+1)} = \omega^{(t)}$
5. Continue to step 2 unless the termination condition is met

There is another approach to sampling the Markov chain called *Metropolis -Hastings*[17][18]. The difference between the BFACF algorithm above with Metropolis-Hastings BFACF is the way to choose a elementary move to perform. In the step 3 of the BFACF algorithm, *Metropolis-Hastings BFACF* is modified as follows:
choose one of the 2d-2 directions uniformly and accept/reject to perform the elementary move. The *Metropolis-Hasting* style BFACF will be discussed in section 4.1

# Chapter 3

# the Wang-Landau algorithm with BFACF moves

## 3.1 Theory

The BFACF algorithm involves three types of moves, +2, -2, +0, which change the length of the conformation (increase by 2, decrease by 2 or the length does not change, respectively). In the figure 1.1, the first row of moves indicates +2 and -2 moves. From left to right, the length of the polygon increases from 3 to 5 and similarly, from right to left the length of the polygon decreases from 5 to 3. The second row of Figure the moves illustrates +0 moves. Either polygon from left to right or the opposite direction, the length polygon remains.

**Remark 3.1.1.** +2, -2, +0 moves do not break the cubic lattice knot and the knot type will be maintained; also, topological invariants like the linking number will be maintained.

Instead of utilizing the definition of distribution from BFACF, Wang-Landau defines the distribution as follows:

**Definition 3.1.1** (Wang-Landau distribution)**.**

$$\pi(\sigma) = \frac{1}{\textit{the number of energy states}} \cdot \frac{1}{\mu^k_{E(\sigma)}} \tag{3.1}$$

Where $\sigma$ is a particular conformation, k is the knot type of $\sigma$ and $\mu^k_{E(\sigma)}$ is the number of energy $E(\sigma)$ conformations of k knot type in $\mathbb{Z}^3$. The Wang-Landau distribution is a uniform distribution.

**Definition 3.1.2** (Wang-Landau assumption)**.**

$$\frac{\mu_{E_1}}{\mu_{E_2}} \approx \frac{e^{w_{E_1}}}{e^{w_{E_2}}} \tag{3.2}$$

$\mu_{E_i}$ is the number of energy $E_i$ conformations, $w_{E_i}$ is the training weight of the energy state $E_i$. In the assumption, the exponential of the weights ratio converges to the density ratio.

**Definition 3.1.3** (Detailed Balance)**.**

$$\pi(\sigma) \cdot P(\sigma \to \sigma') = \pi(\sigma') \cdot P(\sigma' \to \sigma) \tag{3.3}$$

Plug the distribution (3.1) into the detailed balance equation(3.3):

$$\begin{aligned} &\frac{1}{\textit{the number of energy states}} \cdot \frac{1}{\mu^k_{E(\sigma_i)}} \cdot \frac{1}{|\sigma_i|} \cdot \frac{1}{4} \cdot P_{acc}(\sigma_i \to \sigma_j) = \\ &\frac{1}{\textit{the number of energy states}} \cdot \frac{1}{\mu^k_{E(\sigma_j)}} \cdot \frac{1}{|\sigma_j|} \cdot \frac{1}{4} \cdot P_{acc}(\sigma_j \to \sigma_i) \end{aligned} \tag{3.4}$$

**Remark 3.1.2.** $P_{acc}(\sigma \to \sigma')$ is different than $P(\sigma \to \sigma')$. $P_{acc}(\sigma \to \sigma')$ is the probability of accepting the chosen move by the Metropolis Hastings algorithm, whereas $P(\sigma \to \sigma')$ is the transition probability from state $\sigma$ to $\sigma'$. The transition probability can be represented as the product of the probability of uniformly

choosing a move and the probability of acceptance. In the context of SAPs in $\mathbb{Z}^3$, the probability of uniformly choosing a edge is $\frac{1}{|\sigma|}$ and that of uniformly choosing a direction to push is $\frac{1}{4}$.

Equation 3.4 Simplifies to:

$$\frac{P_{acc}(\sigma_i \to \sigma_j)}{P_{acc}(\sigma_j \to \sigma_i)} = \frac{\mu^k_{E(\sigma_i)}}{\mu^k_{E(\sigma_j)}} \cdot \frac{|\sigma_i|}{|\sigma_j|} \tag{3.5}$$

By the definition of probability: If $\frac{\mu^k_{E(\sigma_i)}}{\mu^k_{E(\sigma_j)}} \cdot \frac{|\sigma_i|}{|\sigma_j|} > 1 \Rightarrow P_{acc}(\sigma_i \to \sigma_j) = 1 \Rightarrow \frac{1}{\frac{\mu^k_{E(\sigma_i)}}{\mu^k_{E(\sigma_j)}} \cdot \frac{|\sigma_i|}{|\sigma_j|}} < 1 \Rightarrow P_{acc}(\sigma_j \to \sigma_i) = \frac{1}{\frac{\mu^k_{E(\sigma_i)}}{\mu^k_{E(\sigma_j)}} \cdot \frac{|\sigma_i|}{|\sigma_j|}}$

Substitute the Wang-Landau Assumption(3.2):

$$P_{acc}(\sigma_i \to \sigma_j) = min(\frac{e^{w_{E_i}}}{e^{w_{E_j}}}, 1) \tag{3.6}$$

## 3.2 Training algorithm

The simplified Wang-Landau Algorithm estimates the density of energy of a system as follows:

1. Specify a modification factor $f$, initial density estimate $\alpha$, an initial state $\sigma$ and a terminate condition.

2. Perform some number of Markov steps using

$$P_{acc}(\sigma_i \to \sigma_j) = min\{1, \frac{|\sigma_i|}{|\sigma_j|} \cdot \frac{e^{w_{E_1}}}{e^{w_{E_2}}}\} \tag{3.7}$$

   for the Metropolis-Hastings step.

3. Update $w_{E_1}$ and $w_{E_2}$

4. Return to step 2 unless the termination condition is met.

**Detailed explanation**:

**NOTE:** For the purpose of optimizing computation, the weights $w_E$ and the increment $f$ are modified to logarithm. The weights stored in a weights list are the exponential of $e$ and $f$ is added to weight instead of production. This helps keep the scale of the weights small.

1. Specify a modification factor $f$, initial density estimate $\alpha$, an initial state $\sigma$ and a terminate condition.

   - The modification factor $f$ is the increment to weight specifically. Each time the chosen step is been performed, the application adds the $f$ to the associated state.
   - The initial density estimate is the distribution of substrates. If there is only one substrate been fed, the initial density estimate $\alpha$ is been set to 0 for ever energy state, as no state has ever been before. If the substrates have more than one conformation, then the initial density estimate will be the portion of the element in the sample space.
   - The initial state is the initial conformation that the algorithm starts with.
   - Termination condition is the condition that tells the application when to stop. In theory, the Wang Landau algorithm stops when it discovers almost the whole space, however this is hard to assess in a numerical experiment. Thus, a good way to do it is to set a threshold for $f$ because $f$ is the increment of the weight. When $f$ small enough, the weights are stable.

2. Perform some number of Markov steps using equation(3.7)

- Specifically, the Markov steps are the following: choose an edge uniformly → choose a direction uniformly → decide whether accept or reject by equation 3.7 → check Self-Avoidance → perform the chosen move or stay.
- The appropriate number of steps before updating the weight is not predetermined. In this implementation, the experimental result support an improvement on convergence speed of training by allowing more steps before updating the weight list, and we will discuss in section 4.3

3. Update $w_{E_1}$ and $w_{E_2}$
   - It is important to update the current state when rejected or when chosen move is self-Avoiding. For the purpose of the convergence of the Markov Chain, the irreducibility needs to be satisfied. Also, it can help the chain escape the "trapped" area. In the Markov process, the chain may fall into a favorable area can stay there for a long time. However, the chain staying in the favorable area will decrease the portion of samples that are in the unfavorable area. If we keep increasing the weights for the favorable area by equation 3.7, the chain will gain more probability to escape.
   - The timing for the weight update is also important. As mentioned above there is not best answer for the number of Markov steps before the update. In the classic Ising model, the algorithm updates the weight after every move. However, through the test that using different number of steps before updates, the results converge faster with around 1000 steps the before updating the weights.
4. There is an implicit terminate condition in order to reduce the value of $f$. After some amount of weights updating, the algorithm will check if it has

sampled the whole space. There will be an other list called"histogram" that keeps tracking the occurrence of the conformations with a specific $f$. The histogram will be cleared every time reducing $f$. At each $f$, the histogram tells how many times each state has been visited. When the histogram is getting "flat", we say that the algorithm has mostly discovered the whole space. The flatness criteria for the histogram will be discussed in section 4.2.

5. Check if $f$ is small enough and then terminate the sampling process.

## 3.3 Sampling algorithm

The sampling algorithm utilizes the approximation of the distribution obtained from the training process. The training process provides a weight list that the ratio of the weights approximates the relative ratio of densities with corresponding energy states. Thus, along with the sampling process, the acceptance probability is calculated at each step by equation 3.7. For example, rare conformations in the sample space have smaller weights compared to common conformations. Thereby, if the chosen move is from a common conformations to a rare conformations, the probability of acceptance is close to 1, which means it will be very likely accept the move. However, the training weight has similar values for conformations within the neihborhood(i.e. related by just few BFACF moves). We need to choose an appropriate step size to reduce the correlation of samples. The general *sampling algorithm* is described as follows:

1. Input training weight list, initial polugon and specify the step size $s$ and the number of samples $n$;

2. Perform $s$ Markov steps by equation 3.7;

3. Write out current sample to file;

4. Return to step 2 unless the number of samples exceeds $n$.

The correlation can be realized after sampling. Thus, the user has to increase the step size $s$ after the sampling process to assure the samples are mostly independent to each other.

# Chapter 4

# Results and Comparison to the BFACF algorithm

## 4.1 Metropolis-Hastings BFACF

The Arsuaga Vazquez lab has used the BFACF algorithm in a variety of studies[19]. In this work we propose two improvement to the classic BFACF algorithm. First we implement BFACF with *Metroplis-Hastings* sampling. Second, we implement BFACF moves with Wang Laudau sampling. The *Metropolis-Hastings* style BFACF has direct adjustment to the probabilities of each BFACF moves. Our *Metropolis-Hastings* style BFACF used the probabilities in equation 2.14 for the acceptance probabilities at first implementation attempt. We expect the result to converge to an ensemble of polygons with similar average length as with classic BFACF algorithm. However, the probabilities in equation 2.14 do not result in an optimal choice for elementary moves' probabilities. Appropriate samples should show independence with each other can be measured by autocorrelation indicating the internal correlation of sequential sampling. With the standard choice equation 2.14, it is trivial that the acceptance probabilities of $-2$ and $+0$

moves are strictly less than 1, satisfying the sum of all possible moves' probabilities less or equal than 1. Therefore, the $-2$ and $+0$ moves can be rejected where the conformation remains and therefore, increasing autocorrelation in sequential sampling. In the method of *Metropolis-Hastings* style BFACF, instead of choosing the elementary moves with probabilities as in equation 2.14, a move is chosen first so that the choice does not constrain the sum of acceptance probabilities of all possible moves. In order to reduce correlation, we can increase the acceptance probabilities of $-2$ and $+0$ moves to 1, where the algorithm will always accept $-2$ and $+0$ moves(instead of rejecting and remaining the same conformation). For satisfying equation 2.13, the acceptance probability of $+2$ move set to $z^2$.

**Definition 4.1.1.** The modified acceptance probabilites is defined as follows:

$$\begin{aligned} p(+2) &= z^2 \\ p(-2) &= 1 \\ p(+0) &= 1 \end{aligned} \tag{4.1}$$

In the simulation, the step size is the frequency of sampling. 10,000 step size indicates the simulation draw a sample every 10,000 BFACF moves, similarly for the 1 million and 5 million step size. Increasing the step size helps independence and as the autocorrelation approaches 0.5, the simulation converges to the desired distribution. From table 4.1, increasing the step size from 10,000 to 5 million for both standard choice and modified choice reduce the autocorrelation to $\sim 0.5$. Also, as we increase the step size, for both standard choice and modified choice, the error of the mean length goes down. From standard choice to modified choice, autocorrelation and error for the mean length go down for all step sizes. Since the simulation use the same sequence of seed, we can compare the data output line by line.

Table 4.1: Autocorrelation and Mean length with respect to different step size

| | Autocorrelation | Mean Length |
|---|---|---|
| 10k | 70.36 ± 28.19 | 124.30±7.39 |
| 10k* | 55.67±19.01 | 124.93±6.85 |
| 1M | 1.22±0.08 | 125.02±1.03 |
| 1M* | 0.91±0.05 | 124.89±0.88 |
| 5M | 0.56±0.03 | 124.95±1.05 |
| 5M* | 0.53±0.02 | 124.79±0.68 |

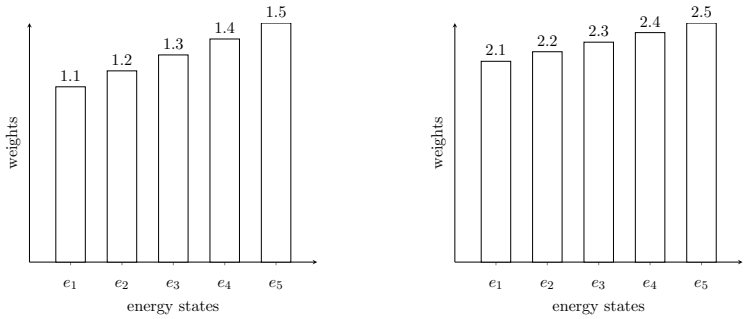
Cooperation of autocorrelation and mean length with respect to 10,000 step size, 1 million step size and 5 million step size. 10k, 1M, 5M represent 10,000 step size, 1 million step size, 5 million step size respectfully sampling with standard choice probabilities. Entries indicated with * represent sampling with modified probabilities. For each step size, there are 100 repetitive simulations with unique seed(we use the same 100 unique seed for different step size simulations) and each simulations sampling 1000 conformations. We use an integrated autocorrelation function (code courtesy of profession student whittington, University of Toronto), where a value 0.5 indicates no autocorrelation and thus is a moxy for an independent sample.

## 4.2 Wang Landau simulation: flatness check criteria

In the training phase, the algorithm maintains a histogram that counts the observation of different energy classes. The flatness check provides a measure of how well the algorithm understands the sample space, i.e are there enough observation for each energy class? The naive way to start is by calculating the min/max value of histogram and setting a threshold to measure the flatness of the histogram. The implementation starts with this naive min/max value ratio criteria and tests the performance of these criteria. The min/max value ratio criteria work to ensure the completeness of the algorithm for knot with relative simple topology. However, for complicated knots the ratio criteria lacks adequate performance.

The problem of min/max value is that for a particular weight list increment, there exists a limit for the accuracy of pairwise weights. The ratio criteria cannot sense if the current increment, reach the desired accuracy. In order to satisfy the

conditions imposed by ratio criteria, the algorithm will waste time on increasing every energy states' weight. In the following figures, if the flatness criteria is been set to min/max $\geq 0.8$ the plot in Fig.4.1a does not pass the criteria whereas the one in Fig. 4.1b passes the criteria. However, the differences of weights are same for Fig. 4.1a and Fig. 4.1b. For example, $w_{e_2} - w_{e_1} = 0.1$ for both plots, and similar to other pair of energy state such as $w_{e_4} - w_{e_2} = 0.3$. In the Eq. 3.6 we calculate acceptance probability by $e^{w_{E_i} - w_{E_j}}$ so the 4.1a already satisfies the expectation for starting weight list and the 4.1b does not give us improved accuracy.




(a) before (b) after

Figure 4.1: Example of checking the flatness by ratio test

Specifically, the Wang Landau algorithm starts with a rough weight list and keeps perfecting the weight list by updating weight list while walking in the sample space. The extreme strict criteria affect the correctness of the algorithm, because the extreme strict criteria force all energy states to be visited evenly where the weight list and histogram end up with horizontal lines. Horizontal lines mean energy are uniformly distributed, which is not true. Thus, we should not check flatness in extreme strict otherwise it will destroy the approximation of the sample space.

For this reason, our current solution is to use a loose criteria at the boundaries:

we only require some extent of time of observation on observed energy states and normalize the weight list after the flatness criteria have been passed. This solution is supported by the reason of the walk intend to expand unvisited energy state initially. Unvisited energy state has 0 weight value and by Eq. 3.6, this move will be accept with probability 1. As the initial state is set to minimal energy state, the criteria requires at least 2 times observation on the initial state for the first flatness criteria. Normalization of the weights list helps the computation directly, since the actual value of the weight is meaningless, by Wang Landau assumption 3.2 the difference of $w_{E_i}, w_{E_j}$ in $e^{w_{E_i} - w_{E_j}}$ is the crucial value. Since to the first flatness check is the first time exploring the sample space, passing the flatness check is a time consuming process. Thus, the solution only require 2 observations for the first flatness check, but more observations in future flatness checks. We will discuss more flatness check criteria in the section 5.0.1.

## 4.3 Wang Landau simulation: frequency of updates

Frequency of updates refers to how often the weight list and histogram will be updated. In terms of the theory, the algorithm could be set to updating the weight list and histogram after each step. However, in terms of input and output(i.e. I/O corresponds to write/read that consumes computation power) of computation, also, considering the randomness of the Markov Chain, we suppose that the appropriate larger gap between updates helps the speed of convergence. This argument is supported by experiments in Fig. 4.3 and Fig. 4.2 but it lacks theoretical support. The larger the gap between updates means the algorithm will walk some amount of distance in the sample space before updating and it potentially helps to explore the sample space faster.

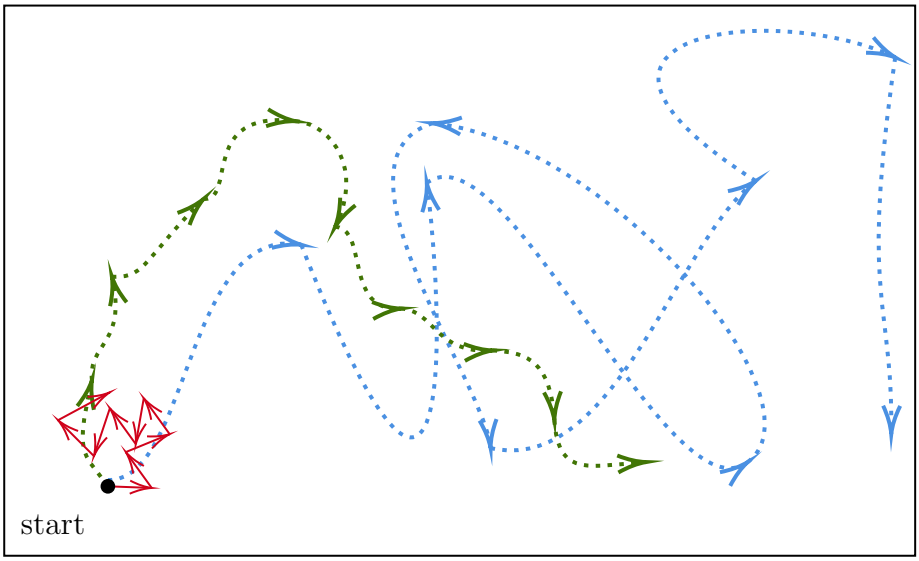



Figure 4.2: Visualization of training with different numbers of Markov steps

The graph showing 3 step size samplings in red, green and blue. Red walk represent update the histogram and weight list at every step, green walk represent medium range step size to update histogram and weight list and blue walk represent the largest step size to update the histogram and weight list. The larger step size helps the walk realize global sample space where smaller step size limits the walk realize local sample space. However, larger step size consume more computation power at each update and it reduce the speed of sampling process directly.

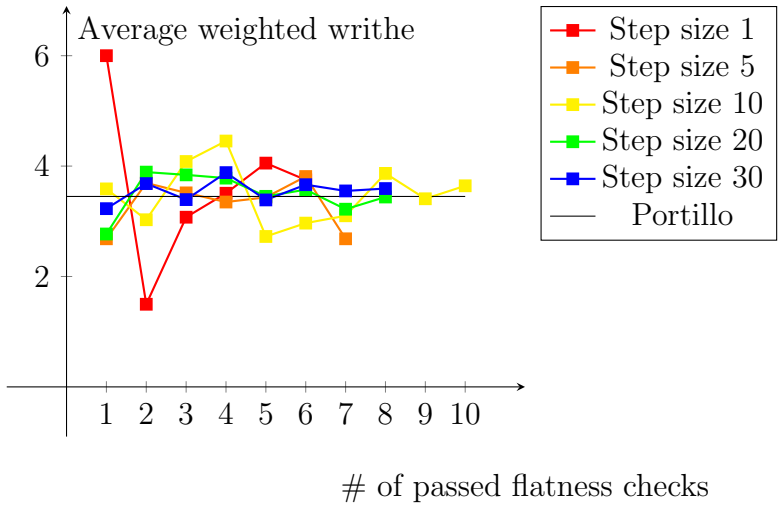
We computed the average weighted writhe to different length for $3_1$, $6_2$ and $7_2$. The average weighted writhe is calculated based on training phase.

**Definition 4.3.1.** Average weighted writhe is defined as follows:

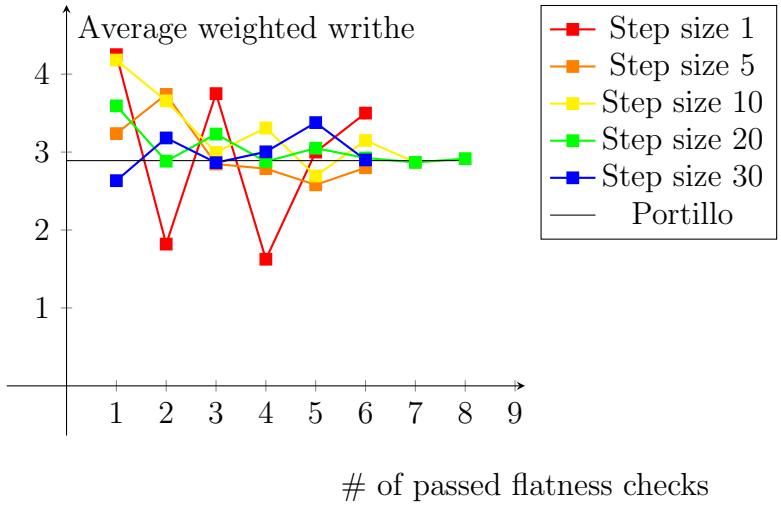
$$m_L = \frac{\sum_{E_l \in L} e^{w_{E_l}} \cdot \mathbf{Wr}_{E_l}}{\sum_{E_l \in L} e^{w_{E_l}}} \tag{4.2}$$

$L$ is the set that energy states that the length is expected to calculate the mean writhe, $w_{E_l}$ is the weight for the energy state $E_l$ and $\mathbf{Wr}_{E_l}$ is the writhe for the energy state $E_l$. We run simulation with step size 1, 5, 10, 20, 30 for knot $3_1$, $6_2$ and $7_2$ at length 74. We compare the results to Portillo et al. BFACF

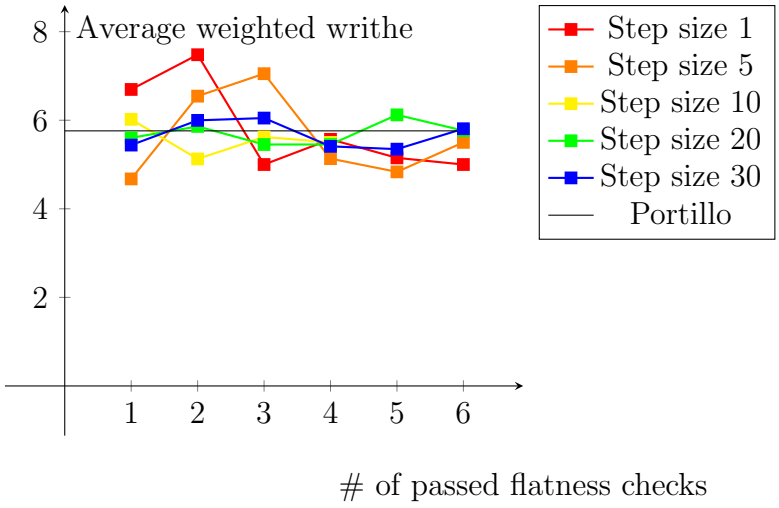
simluation result[19]. Portillo et al. provides the mean writhe of cubic lattice knot at length 75, which at length 75 means the mean length is about 75 because the length of cubic lattice knot has to be even number. As the 4.2 shows, for step size 10, the plot shows it converges faster than step size 1 and has less variance in all plots. However, for $3_1$ the average weighted writhe derived by using step size 10 is not as accurate as using step size 20 and 30, even using step size 20 and 30 passes less flatness checks. Meanwhile, for the knot $6_2$, using step size 20 has the fastest convergence and passes the most flatness checks. Meanwhile, Fig. 4.2 shows the running time for different step sizes to pass flatness check. Therefore, the optimal results should consider both convergence and time consumption to derive overall best sampling results.




(a) Average weighted writhe of knot $3_1$ training with different step size



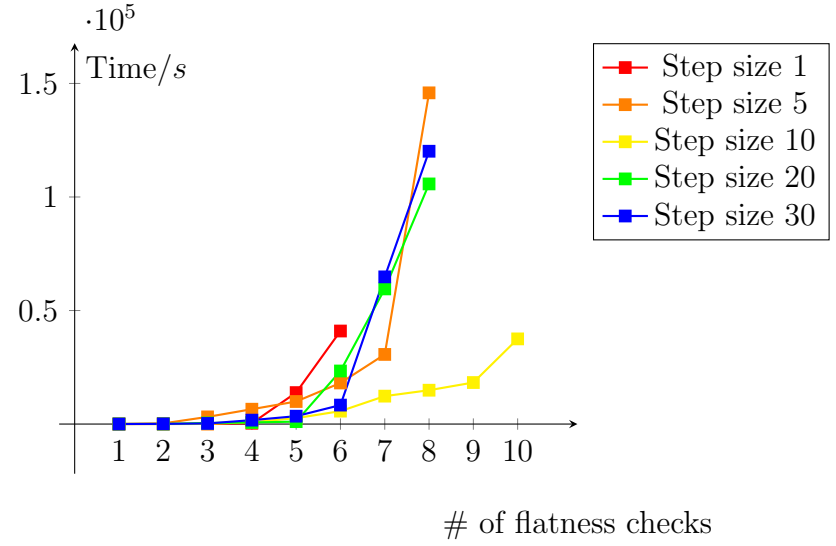

(b) Average weighted writhe of knot $6_2$ training with different step size



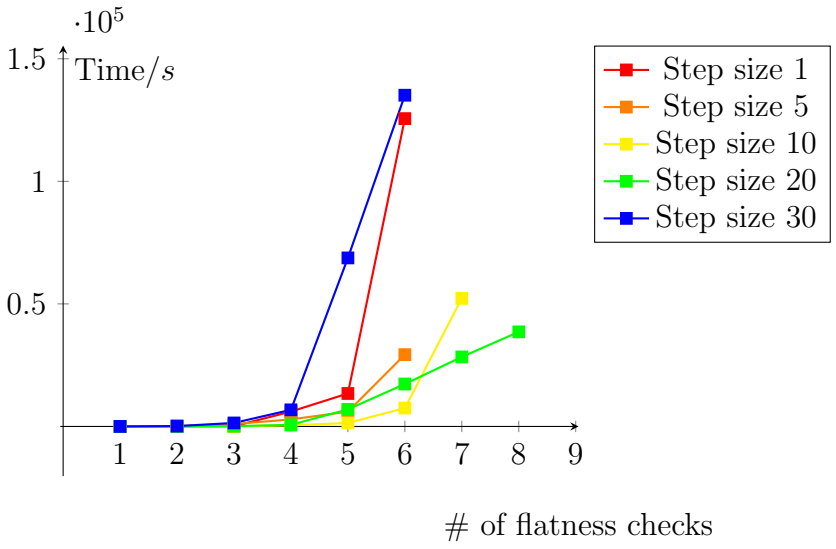

(c) Average weighted writhe of knot $7_2$ training with different step size

Figure 4.2: Average weighted writhe training with different step size for knot type $3_1$, $6_2$ and $7_2$

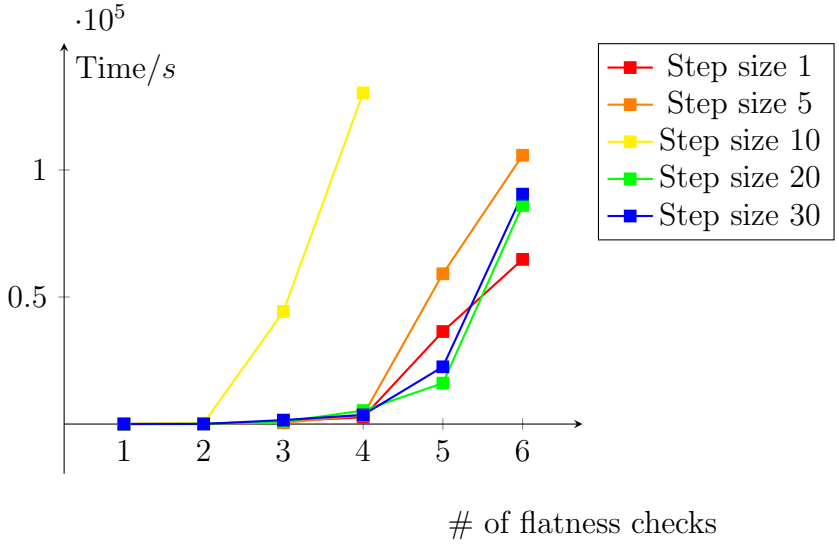
The simulation start at the same time and the more crossing of the knot the simulation takes longer time. For different step size, the maximum numbers of flatness check are different. As we expected, update every step (step size is 1) is not the most efficient frequency to update the histogram and weight list. All knot $3_1$, $6_2$ and $7_2$ average weighted writhe plot shows step size 1 has the largest variance and passes least amount of passed flatness checks.




(a) $3_1$ running time for different step sizes




(b) $6_2$ running time for different step sizes




(c) $7_2$ running time for different step sizes

Figure 4.2: Running time of different step sizes to pass each flatness check for knot type $3_1$, $6_2$ and $7_2$

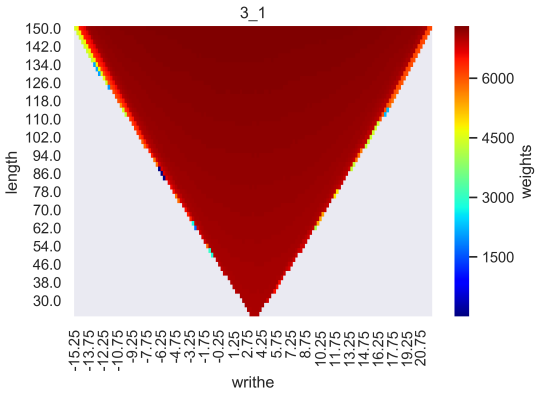
The figure shows that the performance of specific step size varies among different knot types, but in general the larger step size consume more time then lower step size. Note that for Fig. 4.3a step size 1 there is no data after passing 6th flatness check, which means it takes more time to pass 7th flatness check than all other step sizes

## 4.4 Wang Landau simulation: combined Energy and Soft Cut

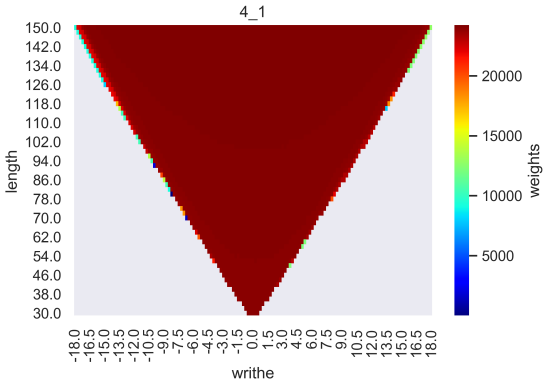
Single energy training and sampling is implemented above, so the goal now is to enable training and sampling of multiple energies. For numerical single energy, the energy state can be represent numerically(integer or float point representation). However, when we combine two or more energies into a state, even energies combined are numerical, the data structure mapping to the combined energies has to be injective, so that the algorithm could update unique energies combination. In addition, if the energy is not restricted to numerical value (String, Char,

Class, Struct, etc.), then a special hash function needed to handle with different types of the input energy combinations.

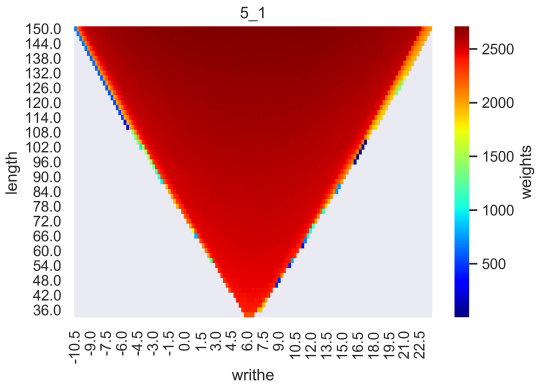
Preferred method is define a customized hash function that handle with conflicts. For simplicity without decrease significant efficiency, the current implementation combines all energy as a string and it formats the energy begin with the capital character of the name of the energy. For example, length 30 with writhe -0.25 is formatted to -0.250000l30. In computer science language, hashing is the function mapping input values into memory location. Hashing string type data into *hashmap* data struction is not the best choice, but the speed of hashing string is reasonably fast and meets all requirements.



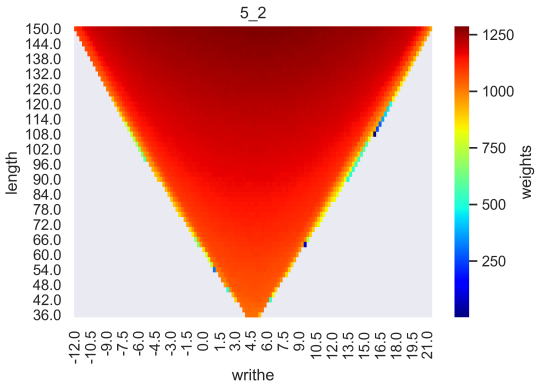

(a) $3_1$



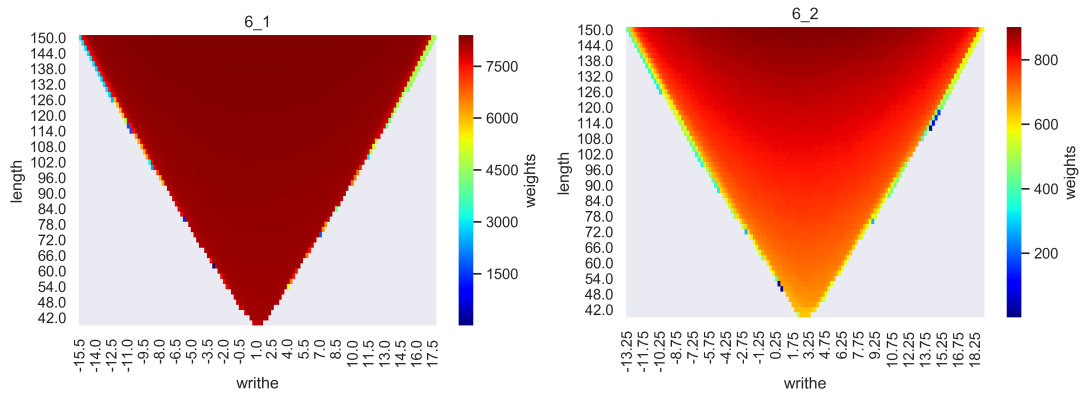

(b) $4_1$




(c) $5_1$




(d) $5_2$




(e) $6_1$ (f) $6_2$

Figure 4.2: Wang Landau training with respect to combined energy length and writhe to different knot types

This figure shows the weights output of the Wang Landau training phase for different knot types. The x axis represents the writhe and the y axis represents the length. The color scale represents the weights of the associated energy state. Larger weights(red) indicate that the associated energy state is more common than the energy state with lower weights(blue) in the sample space. The gray area indicates there is no defined conformation in corresponding energy states.

There is an issue with irreducibility of the Markov Chain when users define the upper bound for the energies in training phase. Training with multiple energies leads to a sample space that is multi dimensional, and the upper bound of length might block paths that the chain walk to a rare energy state.

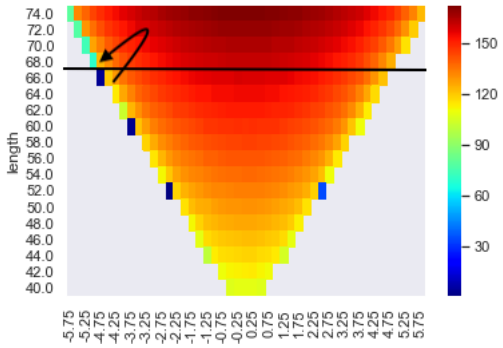



Figure 4.3: Cutting ergodicity in the training weights

The black arrow indicates an example (writhe=-4.75, length=66) of a low energy state(dark blue) that is rare in the sample space. If the user sets the maximum length at 66, it will cut the ergodicity that the chain could go above length 66 and walk back to the energy state (writhe=-4.75, length=66) shown as the arrow in the figure.

The cutting ergodicity situation not only decrease the accurary around the boundary, it also slows down the sampling process of the simulation. With cut ergodicity, one energy state might become less common, which will require more observations at this specific energy state. Since some paths are blocked, the maximum length boundary prevent the chain goes to the energy state less efficiently by reducing number of paths to the energy state, slowing down the sampling process.

## 4.5 Wang Landau simulation: compares to BFACF result

In this section we compares the sampling results obtained from Wang Landau simulation to the results generated by BFACF algorithm from Portillo et al.[19]. The results from Wang Landau simulation is training all knot types up to 8

crossing with maximum length 150 with respect to energy combined length and writhe. The data for Wang Landau simulation displayed in table 4.2 is the average weighted writhe by Eq. 4.2, and we displayed the highest flatness check result from sampling process(the highest number of flatness check is displayed inside parenthesis). Note that the data from Wang Landau simulation is at exact length, for example, we only calculate the average weighted writhe for energy state that length is exactly at 74, 76, 100 or 150. Portillo et al. derive by BFACF algorithm where the mean length of conformations is at 75, 100, 150. There are rows in Portillo et al. data left empty, for example $4_1$, $6_3$, because those knot types share a property that the average writhe should be 0. Compares the data from Wang Landau simulation and BFACF, most of the average weighted writhe does not fall into error bar, but it is highly correlated. It could because the data from Wang Landau simulation is at exact length, but more importantly the Wang Landau simulation need more time to converge. The data shows Wang Landau simulation passing more flatness checks helps the average weighted writhe converge to mean writhe from Portillo et al. When there is only few flatness check passed the deviation is large, for example $4_1$ passing 3 flatness check, at length 74 the average weighted writhe is 0.901 where we expect 0. Also, we could see the average weighted writhe deviates far from mean writhe at length 150. We set the maximum size of the conformations to 150 and it raise the issue of cutting ergodicity.

| | Portillo et al. | | | Wang Landau | | | |
|---|---|---|---|---|---|---|---|
| K | 75 | 100 | 150 | 74 | 76 | 100 | 150 |
| $3_1$ | 3.45 ± 0.03 | 3.49 ± 0.03 | 3.47 ± 0.04 | 3.643(10) | 3.461(10) | 3.5(10) | n/a |
| $4_1$ | n/a | n/a | n/a | 0.901(3) | 0.622(3) | 0.376(3) | 0.184(3) |
| $5_1$ | 6.32 ± 0.02 | 6.31 ± 0.03 | 6.29 ± 0.04 | 5.997(4) | 5.714(4) | 7.15(4) | 8.569(4) |
| $5_2$ | 4.57 ± 0.02 | 4.58 ± 0.03 | 4.61 ± 0.04 | 4.251(4) | 4.393(4) | 4.884(4) | 6.612(4) |
| $6_1$ | 1.17 ± 0.02 | 1.19 ± 0.03 | 1.19 ± 0.04 | 1.409(5) | 1.553(5) | 1.274(5) | 1.303(5) |
| $6_2$ | 2.89 ± 0.02 | 2.88 ± 0.03 | 2.89 ± 0.04 | 3.14(6) | 2.955(6) | 2.95(6) | 3.297(6) |
| $6_3$ | n/a | n/a | n/a | 0.333(4) | 0.769(4) | 0.4(4) | 1.057(4) |
| $6c1$ | 6.96 ± 0.02 | 6.95 ± 0.03 | 6.95 ± 0.04 | 7.255(5) | 7.278(5) | 7.172(5) | 6.497(5) |
| $7_1$ | 9.15 ± 0.02 | 9.14 ± 0.03 | 9.12 ± 0.04 | 9.462(5) | 9.688(5) | 8.965(5) | 7.679(5) |
| $7_2$ | 5.76 ± 0.02 | 5.78 ± 0.03 | 5.82 ± 0.04 | 5.412(6) | 5.401(6) | 5.693(6) | 5.621(6) |
| $7_3$ | 7.40 ± 0.02 | 7.41 ± 0.03 | 7.41 ± 0.04 | 7.125(4) | 6.5(4) | 6.501(4) | 6.816(4) |
| $7_4$ | 5.68 ± 0.02 | 5.68 ± 0.03 | 5.72 ± 0.04 | 5.594(5) | 5.709(5) | 5.195(5) | 5.031(5) |
| $7_5$ | 7.48 ± 0.02 | 7.49 ± 0.03 | 7.52 ± 0.04 | 7.596(5) | 7.417(5) | 6.656(5) | 6.962(5) |
| $7_6$ | 3.40 ± 0.02 | 3.40 ± 0.03 | 3.41 ± 0.04 | 3.754(5) | 3.935(5) | 3.409(5) | 4.874(5) |
| $7_7$ | 0.53 ± 0.02 | 0.54 ± 0.03 | 0.51 ± 0.04 | 0.636(7) | 0.942(7) | 0.753(7) | 0.194(7) |
| $7c1$ | 3.48 ± 0.02 | 3.47 ± 0.03 | 3.50 ± 0.04 | n/a | n/a | n/a | n/a |
| $8_1$ | 2.35 ± 0.02 | 2.37 ± 0.02 | 2.40 ± 0.04 | 2.32(7) | 2.212(7) | 2.47(7) | 3.247(7) |
| $8_2$ | 5.70 ± 0.02 | 5.70 ± 0.02 | 5.66 ± 0.04 | 5.875(7) | 6.125(7) | 5.625(7) | 5.375(7) |
| $8_3$ | n/a | n/a | n/a | -0.25(5) | -0.625(5) | 0.0(5) | 0.542(5) |
| $8_4$ | 1.73 ± 0.02 | 1.71 ± 0.02 | 1.69 ± 0.04 | 1.097(6) | 1.503(6) | 1.62(6) | 1.434(6) |
| $8_5$ | 5.74 ± 0.02 | 5.75 ± 0.02 | 5.71 ± 0.04 | 5.5(4) | 5.125(4) | 6.125(4) | 6.006(4) |
| $8_6$ | 4.07 ± 0.02 | 4.09 ± 0.02 | 4.08 ± 0.04 | 4.0(5) | 3.251(5) | 3.875(5) | 4.614(5) |
| $8_7$ | 2.83 ± 0.02 | 2.83 ± 0.02 | 2.83 ± 0.03 | 2.875(6) | 3.0(6) | 3.25(6) | 3.25(6) |
| $8_8$ | 1.20 ± 0.02 | 1.20 ± 0.02 | 1.22 ± 0.04 | 1.176(6) | 0.697(6) | 0.774(6) | 1.053(6) |
| $8_9$ | n/a | n/a | n/a | -0.057(6) | -0.163(6) | -0.435(6) | -0.595(6) |
| $8_{10}$ | 2.87 ± 0.02 | 2.87 ± 0.02 | 2.87 ± 0.04 | 2.5(3) | 2.625(3) | 3.25(3) | 1.253(3) |
| $8_{11}$ | 3.98 ± 0.02 | 3.96 ± 0.02 | 3.98 ± 0.04 | 4.002(4) | 3.659(4) | 3.434(4) | 2.144(4) |
| $8_{12}$ | n/a | n/a | n/a | 0.682(7) | 0.376(7) | 0.452(7) | -0.671(7) |
| $8_{13}$ | 1.12 ± 0.02 | 1.11 ± 0.02 | 1.12 ± 0.03 | 1.25(5) | 1.375(5) | 0.75(5) | 1.749(5) |
| $8_{14}$ | 4.06 ± 0.02 | 4.07 ± 0.02 | 4.08 ± 0.04 | 4.25(6) | 3.75(6) | 4.0(6) | 4.5(6) |
| $8_{15}$ | 7.99 ± 0.02 | 8.01 ± 0.02 | 8.01 ± 0.04 | 8.033(6) | 8.142(6) | 8.241(6) | 8.162(6) |
| $8_{16}$ | 2.83 ± 0.02 | 2.83 ± 0.02 | 2.83 ± 0.03 | 2.888(4) | 3.314(4) | 3.026(4) | 1.726(4) |
| $8_{17}$ | n/a | n/a | n/a | -0.302(6) | -0.6(6) | 0.142(6) | 0.933(6) |
| $8_{18}$ | n/a | n/a | n/a | 0.066(6) | 0.075(6) | -0.568(6) | 0.006(6) |
| $8_{19}$ | 8.80 ± 0.02 | 8.80 ± 0.03 | 8.78 ± 0.04 | 8.743(6) | 8.901(6) | 8.385(6) | 8.597(6) |
| $8_{20}$ | 1.92 ± 0.02 | 1.92 ± 0.03 | 1.95 ± 0.04 | 1.654(5) | 1.355(5) | 2.081(5) | 2.103(5) |
| $8_{21}$ | 4.68 ± 0.02 | 4.70 ± 0.03 | 4.73 ± 0.04 | 4.495(5) | 4.559(5) | 4.772(5) | 4.739(5) |
| $8c1$ | 9.80 ± 0.02 | 9.79 ± 0.03 | 9.79 ± 0.04 | 9.625(6) | 9.625(6) | 9.5(6) | 10.874(6) |
| $8c2$ | 2.84 ± 0.02 | 2.86 ± 0.02 | 2.82 ± 0.04 | n/a | n/a | n/a | n/a |
| $8c3$ | 8.04 ± 0.02 | 8.06 ± 0.03 | 8.06 ± 0.04 | 8.28(5) | 8.092(5) | 8.486(5) | 8.657(5) |

Table 4.2: Wang Landau simulation training output compares to the BFACF algorithm

# Chapter 5

# Conclusions and future work

The motivation for this thesis is to solve the problems arise from sampling with the BFACF algorithm. These problems are the BFACF algorithm cannot provide adequate samples away from average length and the algorithm is restricted to length driven sampling.

We have propose a new Wang Landau algorithm is used to sample lattice polygon related by BFACF moves. The simulation can explore the low probability areas where the BFACF algorithm could not generate sufficient many samples. This provides improved information on the sampling space. Also, as we showed in Chapter 4, the simulation can sample with respect to writhe and combined energy (length,writhe). Sampling with respect to all energies (not only numerical, also categorical) and combined energies allow users to explore the sample space with higher dimensional perspectives. For example, suppose there is a categorical energy $E_c$ split into 5 states, then one can sample with respect to (length, writhe, $E_c$)

The Wang Landau algorithm successfully enhances the computational simulation of SAP in $\mathbb{Z}^3$ by providing more comprehensive tools to sample uniformly the space of possible conformations of fixed knot type and varying properties such as length and writhe. The implementation could help us study DNA behaviour

with respect to energies, advancing future research.

### 5.0.1 Limitation of the work

The main limitation of the work is that the simulation did not meet the terminating condition in Eq.4.4, and as we discussed in 4.3, we need better choices of step size to each different knot types. Alternatively, we compute the weights list after each flatness check passed. Also, the inaccuracy of the weights around user defined boundaries is another limitation of this work. The inaccuracy near boundary is caused by cutting the ergodicity of the Markov Chain by setting the maximum length of the conformations(4.4). However, setting the maximum length is necessary for simulation, because lattice knots in $\mathbb{Z}^3$ could have infinitely long conformations. We believe the weights around the maximum weights have larger errors compare to smaller length energies' weight. Thus, the measurement of confidence level for the weights near user defined maximum length is not specified in this work.

Another main limitation of this work is that the flatness check criteria could be improved. Morozov et al. point out that the flatness criteria should not be user's choices[20]. Our simulation shows drastic speed reduce after passing several flatness check criteria, also it does not has error measurement. Morozov et al. calculate the number of observations at each flatness check that each energy state has to meet, and the number of observations depends on the most probable energy state with defined energy separation and the increment of the weights. Reducing observation for the rare energy state will faster the convergence and we can measure the error with weights and increment. However, in our simulations with combined energies, the energy separation is not defined, also for some energies are categorical such that it has discrete states then the energy separation does not has numerical meaning.

### 5.0.2 Future work

There are many possible directions for future work. For the Wang Landau implementation itself, the flatness check criteria should not be a user choice and we could explore how to measure the energy separation for combined energies and categorical energies. Improved flatness check criteria will increase the accuracy and boost the speed of convergence. Also, another perspective to speed up the Wang Landau implementation is to implement this algorithm in multi-threads. A multi-threads implementation has great potential to speed up the sampling process. A multi-threads implementation can perform multiple random walks simultaneously. The race condition that different threads has different rates performing random walks need to be carefully handled to ensure all threads updated to same weight list and histogram.

Another direction for the future work is to implement customized weight list output, which requires reshaping the uniformly distributions, so that users could use target samples to applications.

# Appendix A

# Wang-Landau suite documentation

## A.1 Key Classes

### A.1.1 metropolisHastingsBfacf

`metropolisHastingsBfacf` is the main class runs training and sampling. The input arguments are following:

- input file for the substrate
- output file name
- Operators

#### A.1.1.1 Operators for metropolisHastingsBfacf

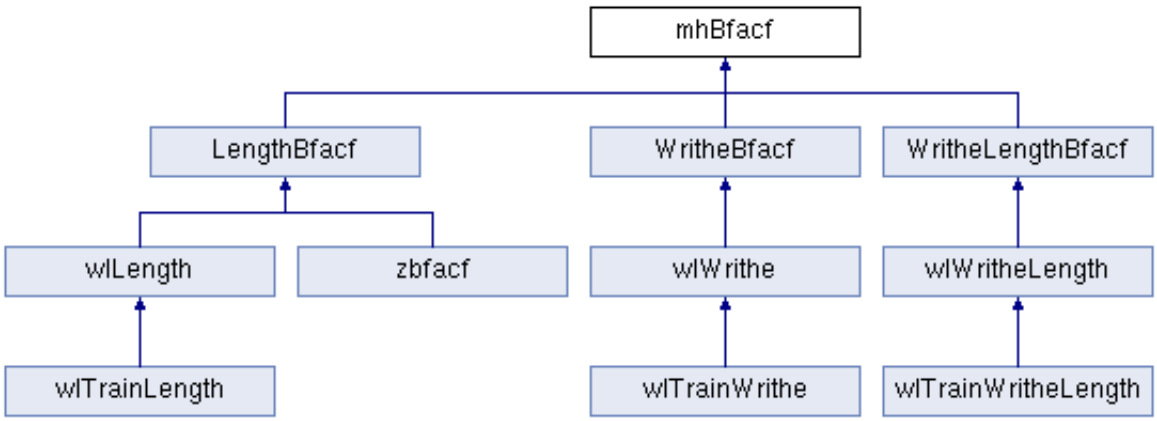
- -mode [s: sampling mode], [t: training mode], [r: BFACF mode]
- -energy the energy for the training or sampling [length: length energy], [writhe: writhe energy], [writheLength: (writhe,length) energy]
- [optional]-low the lower bound for training or sampling
- [optional]-high the upper bound for the training or sampling

- -maxSize the maximum size of the knot/link of training
- -n [required for sampling] the number of the samples
- -f [required for training] the initial weight for training
- -ff [required for training] the final increment to terminate the trainning
- -wlF [required for sampling, optional for retrain] the weights file for sampling or retrain
- -z [required for BFACF] z-value
- -seed [optional] seed for random number generator
- -nStep then step size before update for Wang Landau sampling

**Example A.1.1.** `inputFile outputName -wlF wlFile -mode t -maxSize 100 -f 1 -ff 0.01 -energy writheLength -b 4 -nStep 10`

This is the example of training with input weight list, initial increment is 1, terminate increment is 0.01, training with respect to energy combined length and writhe, step size is 10, and maximum size of conformations is 100

Inheritance diagram for `mhBFACF`:



### A.1.2 mhBfacf

#### Public Member Function

- `mhBfacf (char *inputFile, char *outputFile)`
  super class default consructor.
- `bool add_initial_conformation (char *infile, char *outfile)`
  Initialize the knot from input file
- `void deltaLength (EdgePtr ep, int pushDir)`
  Calculate the difference of length with input push direction at input bond
- `bool checkFlat (int flag)`
  Check the flatness of the histogram. Input flag indicating different flatness check criteria
- `void mhStep ()`
  Choose a random bond and choose a random direction to push
- `virtual void getprob ()=0`
  Virtual function calculate acceptance probability according to class
- `void update (char *energyType)`
  Update the histogram and weight list according to energy type
- `virtual void performMove ()` Virtual function performs a move that contain several Markov steps
- `void train ()`
  Training function performs training algorithm
- `void sample ()`
  Sampling function performs sampling algorithm

### Public Attributes

- `double r`
- `double writhe`
- `double acn`
- `int lowerBound`
- `int upperBound`
- `int maxSize`
- `int buffer`
- `trainSpecs specs`
- `ofstream outputFile`
- `clkConformationAsList initialComp0`
- `clkConformationAsList initialComp1`
- `clkConformationBfacf3 * knot`
- `int n_components`
- `int n_samples`

# References


[1] B. Berg and D. Foerster, "Random paths and random surfaces on a digital computer," *Physics Letters B*, vol. 106, no. 4, pp. 323 – 326, 1981. iv, 2, 3, 20

[2] C. R. Nave, "Statistics of dice throw." `http://hyperphysics.phy-astr.gsu.edu/hbase/Math/dice.html`, n.d. Accessed: 2019-08-31. iv, 13

[3] Goldsim LLC, "Monte carlo simulation what is the monte carlo method?." `https://www.goldsim.com/web/introduction/montecarlo/`, n.d. Accessed: 2019-08-31. iv, 14

[4] E. D. Crawford, S. S. Wilson, K. C. Torkko, D. Hirano, J. S. Stewart, C. Brammell, R. S. Wilson, N. Kawata, H. Sullivan, M. S. Lucia, and P. N. Werahera, "Clinical staging of prostate cancer: a computer-simulated study of transperineal prostate biopsy," *British journal of urology : official journal of the British Association of Urological Surgeons*, vol. 96, no. 7, pp. 999–1004, 2005. 1

[5] C. Lai, H. M. Horlings, M. J. van de Vijver, E. H. van Beers, P. M. Nederlof, L. F. Wessels, and M. J. Reinders, "Sirac: Supervised identification of regions of aberration in aCGH datasets," *BMC Bioinformatics*, vol. 8, p. 422, Oct 2007. 1

[6] Aragão de Carvalho, C. and Caracciolo, S., "A new Monte-Carlo approach

to the critical properties of self-avoiding random walks," *J. Phys. France*, vol. 44, no. 3, pp. 323–331, 1983. 2

[7] C. de Carvalho, S. Caracciolo, and J. Frhlich, "Polymers and $g|\phi|^4$ theory in four dimensions," *Nuclear Physics B*, vol. 215, no. 2, pp. 209 – 248, 1983. 2

[8] E. J. J. van Rensburg and S. G. Whittington, "The BFACF algorithm and knotted polygons," *Journal of Physics A: Mathematical and General*, vol. 24, pp. 5553–5567, dec 1991. 2, 7, 20

[9] F. Wang and D. P. Landau, "Efficient, multiple-range random walk algorithm to calculate the density of states," *Phys. Rev. Lett.*, vol. 86, pp. 2050–2053, Mar 2001. 4

[10] F. Wang and D. P. Landau, "Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram," *Phys. Rev. E*, vol. 64, p. 056101, Oct 2001. 4

[11] W. B. R. Lickorish, *An introduction to knot theory*. Graduate texts in mathematics ; 175, New York: Springer, 1997. 7

[12] J. H. White, "Self-linking and the Gauss integral in higher dimensions," *American Journal of Mathematics*, vol. 91, no. 3, pp. 693–728, 1969. 8

[13] C. Adams, *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. American Mathematical Society, 2004. 9

[14] K. Klenin and J. Langowski, "Computation of writhe in modeling of supercoiled DNA," *Biopolymers*, vol. 54, no. 5, pp. 307–317, 2000. 11

[15] "Lecure notes for introductory probability," December 2017. 14

[16] N. Madras and G. Slade, *The self-avoiding walk*. Birkhauser, 2013. 19

[17] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. 23

[18] W. K. Hastings, "Monte Carlo sampling Methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970. 23

[19] J. Portillo, Y. Diao, R. Scharein, J. Arsuaga, and M. Vazquez, "On the mean and variance of the writhe of random polygons," *Journal of Physics A: Mathematical and Theoretical*, vol. 44, no. 27, p. 275004, 2011. 31, 37, 43

[20] A. Morozov and S. Lin, "Accuracy and convergence of the Wang-Landau sampling algorithm," *Physical Review E*, vol. 76, no. 2, 2007. 47

[21] T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*. Berlin, Heidelberg: Springer-Verlag, 2002.