Applying Different Movesets to the Lattice-SAW Model of Tornadic Vortices

Max Figura and Dr. Pavel Bělík – 2022/07/24

1. Introduction

The Self-Avoiding Walk (SAW) is a type of random walk that does not self-intersect or loop. For some practical usages, it is advantageous to constrain the structure to a lattice grid, restricting the possible directions between any two consecutive points and keeping an equal spacing between consecutive nodes. One field which utilises this approach is that of biochemistry, where the shape of a SAW represents a long but folded-up protein [7]. Another application is in meteorology and fluid mechanics, with the SAW showing the shape of a vortex, such as those found in tornadoes [1]. In both cases, the vast configuration space for SAWs must typically be explored by applying successive random transformations, as determined by some predefined moveset, to the SAW. While a number of movesets have been proposed and used within the former context [3][2][4], little exploration has been done into different movesets used on a vortex model.

We implemented a Markov chain Monte Carlo algorithm that uses four different movesets to sample energy values from the lattice-SAW model of tornadic vortices. Our goal is to determine which moveset, or combination of moveset, best samples the configuration space according to given environmental parameters, as measured by the average energy values. Using this information, we also intend to find more of and better understand the minimum-energy configurations for each length of SAW, which as-of-yet do not have very predictable patterns or behavior.

2. Mathematical Background

Define a Self-Avoiding Walk (SAW) of length N as a series of points $w_0, w_1, ..., w_n$ lying on the lattice grid such that for any points w_i, w_{i+1} (where $i < n$), $|w_i - w_{i+1}| = 1$ and for any distinct indices i, j, $w_i \neq w_j$. Define the steps of a SAW as the series of vectors $s_0, s_1, ..., s_{n-1}$ such that $w_{i+1} = w_i + s_i$ for $0 \leq i < n-1$. Note that, due to the constraining of the SAW to the lattice grid, a given step s_i must be a unit vector in a cardinal direction.

Bělík et al. [1] describe the basis behind representing a vortex as a SAW. If we consider the SAW on some velocity field $\mathbf{u}$, each segment between successive points represents the vorticity $\xi = \nabla \times \mathbf{u}$ of an infinitely thin filament. As we scale up the length N of the SAW, it creates a closer and closer approximation of the real-world case of a tornadic vortex.

The kinetic energy of a configuration is given by

$$E = \frac{1}{8\pi} \sum_i \sum_{j \neq i} \frac{\xi_i \cdot \xi_j}{|\mathbf{m}_i - \mathbf{m}_j|},$$

where ξ_i is the vorticity of the segment from w_i to w_{i+1}, and $\mathbf{m}_i$ is the midpoint of that segment. Because of the dot product, the constant magnitude of segments,

and their constraint to the cardinal directions, we can consider a more intuitive form of the calculation:

$$E = \frac{1}{8\pi} \sum_i \sum_{i \neq j} \begin{cases} \frac{1}{|\mathbf{m}_i - \mathbf{m}_j|} & \text{if } s_i = s_j \\ -\frac{1}{|\mathbf{m}_i - \mathbf{m}_j|} & \text{if } s_i = -s_j \\ 0 & \text{otherwise} \end{cases}.$$

This demonstrates that the largest positive contributions to energy come from adjacent points with the same step direction, while nearby points with opposite step direction have the largest negative contribution. On the scale of the entire SAW, this means that long, straight configurations have the highest energy (with completely straight configurations being the highest), and compact configurations in which the SAW "doubles back" frequently have the lowest energy. It is noteworthy, however, that while the contribution from the interaction of two filaments is inversely related to the distance between them, they do still contribute to the total energy as long as they are parallel or antiparallel. This can make sucessive energy calculations tedious as even a localised change in configuration can necessitate a recalculation of most terms of the sum.

We consider this model under a Boltzmann distribution, where the probability p_i of the system being in a configuration with energy E_i is given by

$$p_i = \frac{e^{-\beta E_i}}{Z}, \qquad Z = \sum_{i=1}^{M} e^{-\beta E_i},$$

where β is an environmental parameter inversely proportional to temperature, and Z is the partition function, which ensures that the sum of the probabilities of all configurations is 1. We can use this probability to find the average kinetic energy of the system for a given temperature, or β, using

$$\langle E \rangle = \sum_{i=1}^{M} p_i E_i.$$

However, calculating the exact value becomes increasingly difficult as finding the value of Z requires the enumeration of all SAWs of the given length, and the number of possible configurations grows exponentially with the length N. Instead, we employ the use of a Markov Chain Monte Carlo (MCMC) algorithm to accurately sample the configuration space with each value of β. To ensure this accuracy, however, it is necessary to use a moveset which can transform the system from one state to another without necessarily causing a large change in energy. The pivot algorithm is a moveset which is considered versatile and used commonly for SAW models [5]; it functions by dividing the SAW into two subsections and then applying some rotational or reflective lattice-preserving transformation to one of them, centered at the attatchment point. Unfortunately, in this context, pivot moves lack the subtlety of more local movesets and can come up short [1]. As such, the focus of this paper will be on the exploration of other movesets to be applied to the lattice-SAW, with the goal of establishing a more reliable alternative.

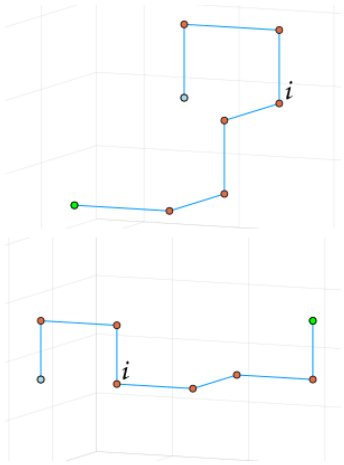


FIGURE 1. Sample pivot move, where i is the "pivot" around which the longer portion of the SAW has been rotated and reflected

3. Movesets Used

We considered four alternatives to the well-established pivot algorithm: Localised Transformations, pull moves, bond-rebridging moves, and Madras transformations. The Localised Transformation (LT) moves were proposed by Bělík et al. in the context of vortices and have been shown to consistently perform better than the pivot moves [1]. The latter three movesets each come from the context of protein folding, designed specifically with the intent of exploring low-energy configurations, and have not been well examined with regards to tornadic vortices. Our goal here is to implement each of the four movesets into this problem and determine if any combination shows an even greater improvement than the LT moves alone.

It must be noted that while LT and pull moves are each fully ergodic for a SAW of a fixed length, rebridge and Madras moves are each unable to explore the full configuration space and therefore cannot be considered in isolation.

3.1. LT Moves.

Localised Transformations (LT moves) were proposed as an efficient and more local alternative to the pivot algorithm. A reconstruction move consists of removing a subsection of the SAW and replacing it with a new, randomly-constructed sub-SAW of the same length. As pointed out by Bělík, this allows for the removal of a "kink" in an otherwise-straight SAW with only one move, by extension resulting in a much less drastic energy change than would be caused by the pivot algorithm. The specific implementation is as follows:

(1) Pick a length l with $1 \leq l \leq n$ from an exponential distribution, such that lower values are more likely.
(2) Pick an index i_1 with $0 \leq i_1 \leq n + 1 - l$. Let $i_2 = i_1 + l$.
(3) Construct a series of replacement steps $s'_{i_1}, s'_{i_1+1}, ..., s'_{i_2-1}$.
(4) Set $s'_j = s_j$ for all $i_2 \leq j \leq n$.

(5) Reconstruct the section of the SAW from w_{i_1+1} to w_{n+1}, setting $w'_{j+1} = w'_j + s'_j$ as j increases from i_1 to n.
(6) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

Note that in situations where $l = n$, the entire SAW is randomly generated anew, demonstrating ergodicity.

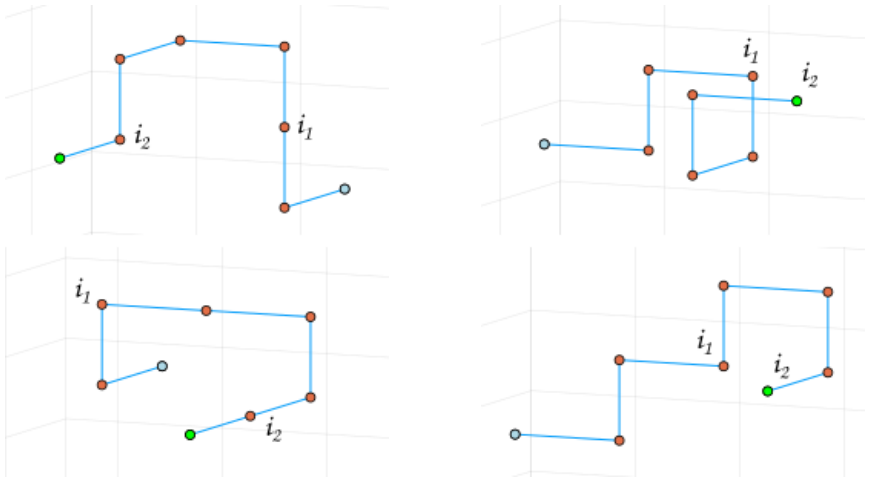
In the interest of further improving efficiency at lower-energy configurations, a special kind of reconstruction move – a permutation move – is used conjointly. A permutation move is identical to a reconstruction move with the specification that the replacement steps $s'_{i_1}, s'_{i_1+1}, ..., s'_{i_2-1}$ must be some permutation of the existing steps $s_{i_1}, s_{i_1+1}, ..., s_{i_2-1}$. In practice, the move is further limited with the specification that the permutation must be a cycle of the steps, taking the form $s'_{i_1}, s'_{i_1+1}, ..., s'_{i_2-1} = s_{i_1+c}, s_{i_1+c+1}, ..., s_{i_2-1}, s_{i_1}, ..., s_{i_1+c-1}$ for some integer c. This is done both for ease of implementation and to reduce the likelihood of self-intersection from cases where $s_j = -s_{j-1}$. Permutation moves also require a somewhat different implementation:

(1) Pick a length l with $2 \leq l \leq K$ (where K is some predetermined integer such that $2 \leq K \leq n$) from an exponential distribution.
(2) Pick an index i_1 with $0 \leq i_1 \leq n + 1 - l$. Let $i_2 = i_1 + l$.
(3) Pick a number of cycles c such that $1 \leq c < l$.
(4) Set $s'_j = s_{j+c}$ for $i_1 \leq j < i_2 - c$ and $s'_j = s_{j-l+c}$ for $i_2 - c \leq j < i_2$.
(5) Reconstruct the section of the SAW from w_{i_1+1} to w_{i_2}, setting $w'_{j+1} = w'_j + s'_j$ as j increases from i_1 to $i_2 - 1$.
(6) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

Using permutation moves in addition to reconstruction moves decreases the proposal probability of moves which displace large sections of the SAW, resulting in a higher success rate in low-energy configurations.

3.2. **Pull Moves.** Pull moves were proposed by Lesh, Mitzenmacher, and Whitesides [3] as another intuitive, localised and ergodic moveset. Each move starts as a displacement of two points along the SAW, but that displacement is propagated back along along one direction of the SAW, "pulling" the rest of the points along with the starting index. To further improve the locality, the propagation of a move will terminate before reaching one end of the SAW if possible. In more specific terms, they use the following process:

(1) Pick an index i with $0 \leq i \leq n$.
(2) If $i = 0$ or $i = n$:
    (a) Set the propagation direction $p = 1$ if $i = 0$, or $p = -1$ if $i = n$.
    (b) Pick two displacement vectors d_1, d_2 to displace the endpoint, creating $w'_0 = w_0 + d_1 + d_2$ and $w'_1 = w_0 + d_1$ (or $w'_n = w_n + d_1 + d_2$ and $w'_{n-1} = w_n + d_1$).
(3) If $0 < i < n$:
    (a) Pick a propagation direction $p = \pm 1$. Set $s = s_i$ if $p = -1$, or $s = -s_{i-1}$ if $p = 1$.
    (b) Pick a displacement vector d orthogonal to s. Set $w'_i = w_i + d + s$ and $w'_{i+p} = w_i + d$.



(A) An LT reconstruction move where $i_1 = 2$ and $l = 4$

(B) An LT permutation move where $i_1 = 3$, $l = 4$, and $c = 2$

FIGURE 2. Sample LT moves

(4) For each point w_{i+pj}, starting at $j = 2$ and with increasing j, set $w'_{i+pj} = w_{i+p(j-2)}$ until $|w_{i+pj} - w'_{i+p(j-2)}| = 1$ or $i + pj = 0$ or $i + pj = n$.
(5) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

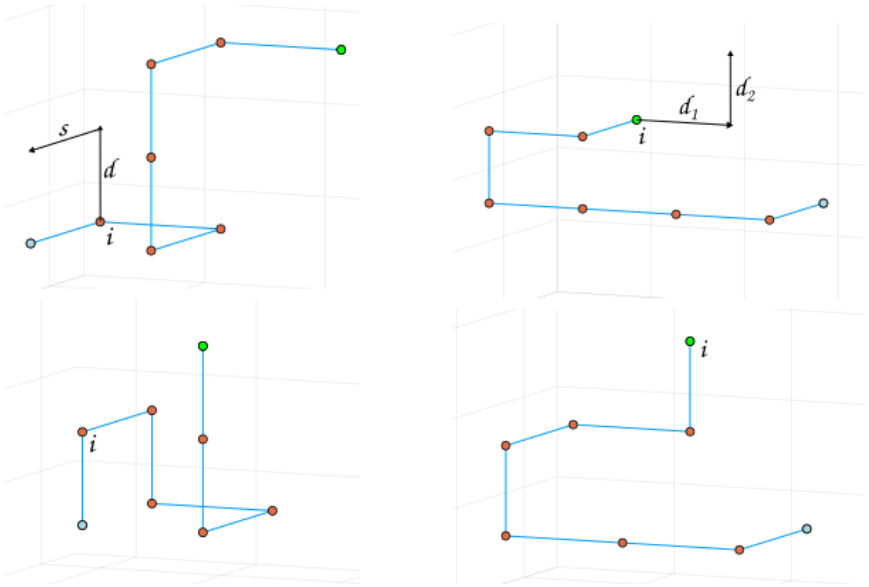
Note that a valid move is assured as long as w'_i and w'_{i+p} are not already visted by the SAW, or w'_i is unvisited and $w'_{i+p} = w_{i+p}$.

It must also be noted that contrary to the claims of Lesh, Mizenmacher, and Whitesides, pull moves as presented in [3] are not fully reversible [8]. Under circumstances where $i = 0$, if $d_2 = s_0$, there exists no pull move that will return to the former location in one step (and similarly for $i = n$). This is a problem for our application, as it disrupts the detailed balance, and by extension, the integrity of the MCMC simulation. However, we were able to show that these particular kinds of end moves (referred to as "hook moves") are the only type of non-reversible pull move (See Appendix A), and they can therefore be prohibited to leave the moveset fully reversible.

Lesh, Mizenmacher, and Whitesides also demonstrate the ability to transform any configuration into a straight line with a finite number of iterations; in conjunction with the (modified) reversibility, this confirms the ergodicity of pull moves.

3.3. **Rebridge Moves.** The bond-rebridging moves were devised by Deutsch [2] in order to better explore the possibilities for very dense and tightly-packed configurations. The rebridging moves do not change the volume the SAW occupies; rather, one will remove the "bond" between two sequential points, then rejoin the resulting sections of the SAW at another location, resulting in a different path through the same space. In practice, there are three distinct types of rebridging moves:

- Single-step rebridges



(A) A pull move where $i = 2$, $p = 1$, and $d = (0, 0, 1)$

(B) A pull move where $i = 7$, $d_1 = (1, 0, 0)$, and $d_2 = (0, 0, 1)$. Note that $|w'_4 - w_3| = 1$, causing the move to terminate early.

FIGURE 3. Sample pull moves

(1) Find a pair of points w_{i_1}, w_{i_2} such that $i_1 + 1 < i_2$, $|w_{i_1} - w_{i_2}| = 1$, and $|w_{i_1+1} - w_{i_2+1}| = 1$.
(2) For $i_1 < j \leq i_2$, set each point $w'_j = w_{i_2+i_1+1-j}$.

- Double-step rebridges
  (1) Find a pair of points w_{i_1}, w_{i_2} such that $i_1 + 3 < i_2$, $|w_{i_1} - w_{i_2}| = 1$, and $|w_{i_1+1} - w_{i_2-1}| = 1$.
  (2) Find a pair of points w_{j_1}, w_{j_2} such that $j_1 \leq i_1$ or $j_1 \geq i_2$, $i_1 < j_2 < i_2$, $|w_{j_1} - w_{j_2}| = 1$, and $|w_{j_1+1} - w_{j_2+1}| = 1$ or $|w_{j_1+1} - w_{j_2-1}| = 1$.
  (3) Create the loop l where

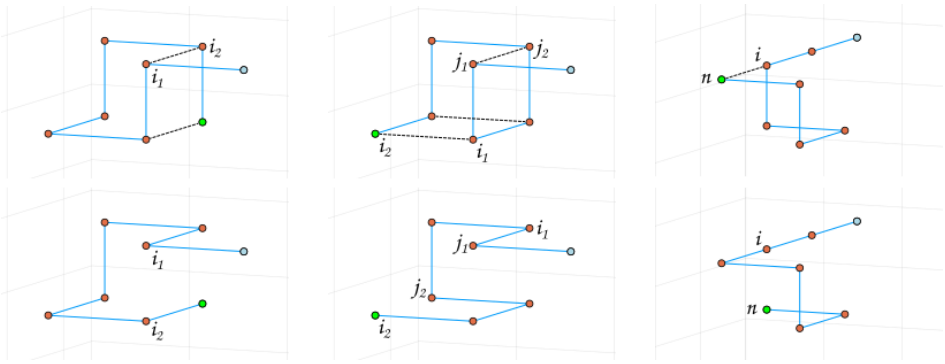
$$l_k = \begin{cases} w_{j_2+k} & \text{if } 0 \leq k \leq i_2 - j_2 - 1 \text{ and } |w_{j_1+1} - w_{j_2-1}| = 1 \\ w_{j_2-i_2+i_1+1+k} & \text{if } i_2 - j_2 \leq k \leq i_2 - i_1 - 2 \text{ and } |w_{j_1+1} - w_{j_2-1}| = 1 \\ w_{j_2-k} & \text{if } 0 \leq k \leq j_2 - i_1 - 1 \text{ and } |w_{j_1+1} - w_{j_2+1}| = 1 \\ w_{j_2+i_2-i_1-1-k} & \text{if } j_2 - i_1 \leq k \leq i_2 - i_1 - 2 \text{ and } |w_{j_1+1} - w_{j_2+1}| = 1 \end{cases}$$

  (4) For $0 \leq k \leq n$, set each point

$$w'_k = \begin{cases} w_k & \text{if } (k \leq i_1 \text{ or } k \geq j_1 + 1) \text{ and } j_1 \geq i_2 \\ w_{i_2-i_1-1+k} & \text{if } i_1 + 1 \leq k \leq j_1 - i_2 + i_1 + 1 \text{ and } j_1 \geq i_2 \\ l_{k-j_1+i_2-i_1-2} & \text{if } j_1 - i_2 + i_1 + 2 \leq k \leq j_1 \text{ and } j_1 \geq i_2 \\ w_k & \text{if } (k \leq j_1 \text{ or } k \geq i_2) \text{ and } j_1 \leq i_1 \\ l_{k-j_1-1} & \text{if } j_1 + 1 \leq k \leq i_2 - i_1 + j_1 - 1 \text{ and } j_1 \leq i_1 \\ w_{i_1-i_2+1+k} & \text{if } i_2 - i_1 + j_1 \leq k \leq i_2 - 1 \text{ and } j_1 \leq i_1 \end{cases}$$

- End rebridges
  (1) Pick an endpoint w_0 or w_n.
  (2) Find a point w_i such that $|w_i - w_0| = 1$ (or $|w_i - w_n| = 1$).
  (3) For $0 \leq j < i$, set each point $w'_j = w_{i-j-1}$ (or, for $i < j \leq n$, set each point $w'_j = w_{n+i+1-j}$).

Note that, instead of at any point checking that a space is empty, we must instead check that a given point related to the move does lie on the SAW. This is because the points occupied by the SAW will remain the same before and after the move. It is thus easy to see how the bond-rebridging moveset is not fully ergodic.



(A) A single-step rebridge move where $i_1 = 1$ and $i_2 = 6$

(B) A double-step rebridge move where $i_1 = 2$, $i_2 = 7$, $j_1 = 1$, and $j_2 = 4$

(C) An end rebridge move with w_n where $i = 2$

FIGURE 4. Sample bond-rebridging moves

3.4. **Madras Moves.** Last are the Madras moves, proposed by Madras, Orlitsky, and Shepp [4]. They serve to spatially transform a subsection of the SAW – reflection across a plane, reflection through a point, or rotation about an a axis – such that it still lies on the cubic lattice and the rest of the SAW is unchanged. For any two indices i_1, i_2 (with $i_1 < i_2$), the three moves are performed as follows:

- Inversion
  (1) For $i_1 \leq j < i_2$, set $s'_j = s_{i_1+i_2-1-j}$.
  (2) Reconstruct the section of the SAW from w_{i_1+1} to w_{i_2}, setting $w'_{j+1} = w'_j + s'_j$ as j increases from i_1 to $i_2 - 1$.
  (3) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

  This has the overall effect of reflecting a portion of the SAW through the midpoint of the line from w_{i_1} to w_{i_2}.
- Reflection
  (1) Find a pair of distinct coordinate dimensions α, β such that $\dfrac{w_{i_2}(\alpha) - w_{i_1}(\alpha)}{w_{i_2}(\beta) - w_{i_1}(\beta)} = m$ for some $m = \pm 1$.
  (2) For $i_1 \leq j < i_2$, set $s'_j(\alpha) = m s_{i_1+i_2-1-j}(\beta)$, $s'_j(\beta) = m s_{i_1+i_2-1-j}(\alpha)$, and $s'_j(\gamma) = s_{i_1+i_2-1-j}(\gamma)$ for $\gamma \neq \alpha, \beta$.

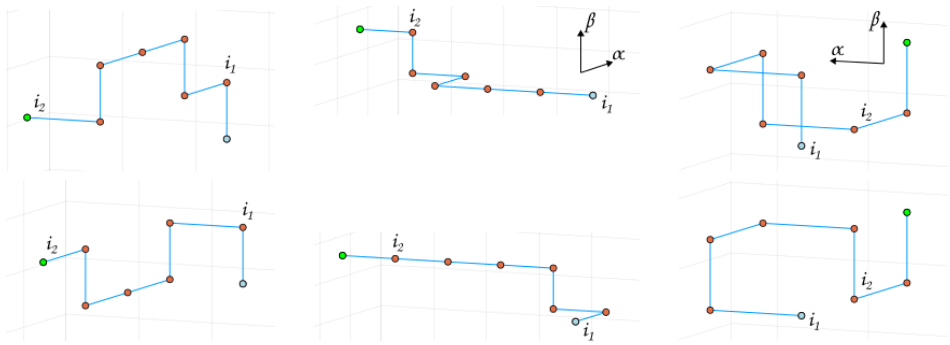
(3) Reconstruct the section of the SAW from w_{i_1+1} to w_{i_2}, setting $w'_{j+1} = w'_j + s'_j$ as j increases from i_1 to $i_2 - 1$.
(4) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

Contrary to the nomenclature, when applied in three dimensions, this move has the effect of rotating a portion of the SAW about a perpendicular bisector (parallel to the $\alpha - \beta$ plane) of the line from w_{i_1} to w_{i_2}.

- Interchange
  (1) Find a pair of distinct coordinate dimensions α, β such that $\frac{w_{i_2}(\alpha) - w_{i_1}(\alpha)}{w_{i_2}(\beta) - w_{i_1}(\beta)} = m$ for some $m = \pm 1$.
  (2) For $i_1 \leq j < i_2$, set $s'_j(\alpha) = m s_j(\beta)$, $s'_j(\beta) = m s_j(\alpha)$, and $s'_j(\gamma) = s_j(\gamma)$ for $\gamma \neq \alpha, \beta$.
  (3) Reconstruct the section of the SAW from w_{i_1+1} to w_{i_2}, setting $w'_{j+1} = w'_j + s'_j$ as j increases from i_1 to $i_2 - 1$.
  (4) If the resulting configuration is self-avoiding, the move is successful; otherwise, the move fails and returns the previous configuration.

  This move has the effect of reflecting a portion of the SAW across the plane through w_{i_1} and w_{i_2} and perpendicular to the $\alpha - \beta$ plane.

The moveset was designed so as to leave the enpoints of the SAW unchanged; it is easy to see why it lacks ergodicity.



(A) A Madras inversion move where $i_1 = 1$ and $i_2 = 7$

(B) A Madras reflection move where $i_1 = 0$, $i_2 = 6$, $\alpha = y$, and $\beta = z$

(C) A Madras interchange move where $i_1 = 0$, $i_2 = 5$, $\alpha = x$, and $\beta = z$

FIGURE 5. Sample madras moves

4. Results

4.1. **Validating Movesets.** In checking the validity of these various movesets, it is first important to make an absolute comparison with known quantities. The full space of configurations – and energy of each – has been enumerated for $N \leq 9$, so the exact average energies can be calculated and the simulated averages checked against

them. For each $N \in \{3, 4, 5, 6, 7, 8, 9\}$, we ran a series of trials from $\beta = -100$ to $\beta = 100$, with $\Delta\beta = 0.5$. In each trial, we start with a completely straight, vertical configuration and attempt $10N$ moves at $\beta = 0$ to obtain a suitably unrelated configuration. Then, 100,000 burn-in moves are attempted at the desired β value to settle in to that temperature. Finally, 200,000 moves (300,000 moves for the longer-to-settle values $-40 < \beta < -5$ and $30 < \beta < 100$) are attempted while keeping a running average of the resulting energy values. The final average for each trial is recorded and plotted as a point in Fig. 6 below; underneath the points lies a curve representing the exact values.

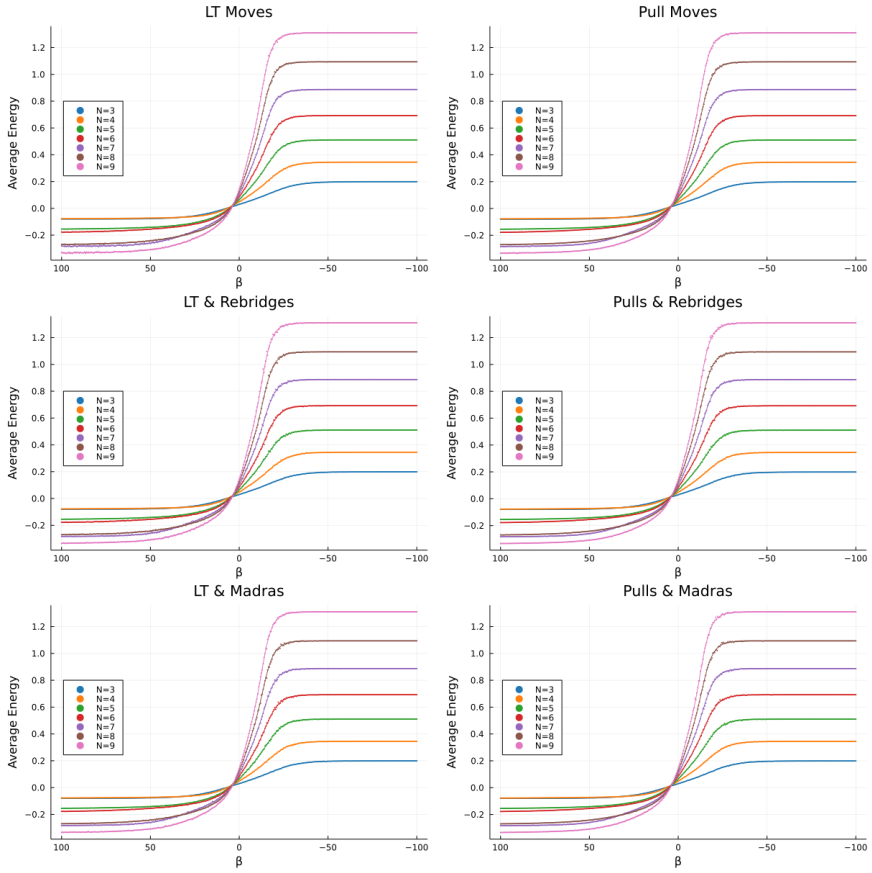


FIGURE 6. Simulation-generated average energies for $3 \leq N \leq 9$, $-100 \leq \beta \leq 100$ (points) overlaying exact values (curves). The β axis is reversed so as to correspond with an increasing temperature from left to right.

It's clear to see that each combination of movesets tested does conform fairly well to the calculated values. As shown in Fig. 7, they are all consistently off in both directions, impying that there is no systematic error present in any moveset. An exception to this is for large negative values of β, where all simulations converged to the maximum energy value quicker than the exact values did; the discrepency is rather small and for our purposes can be ignored. Each moveset is most unstable for

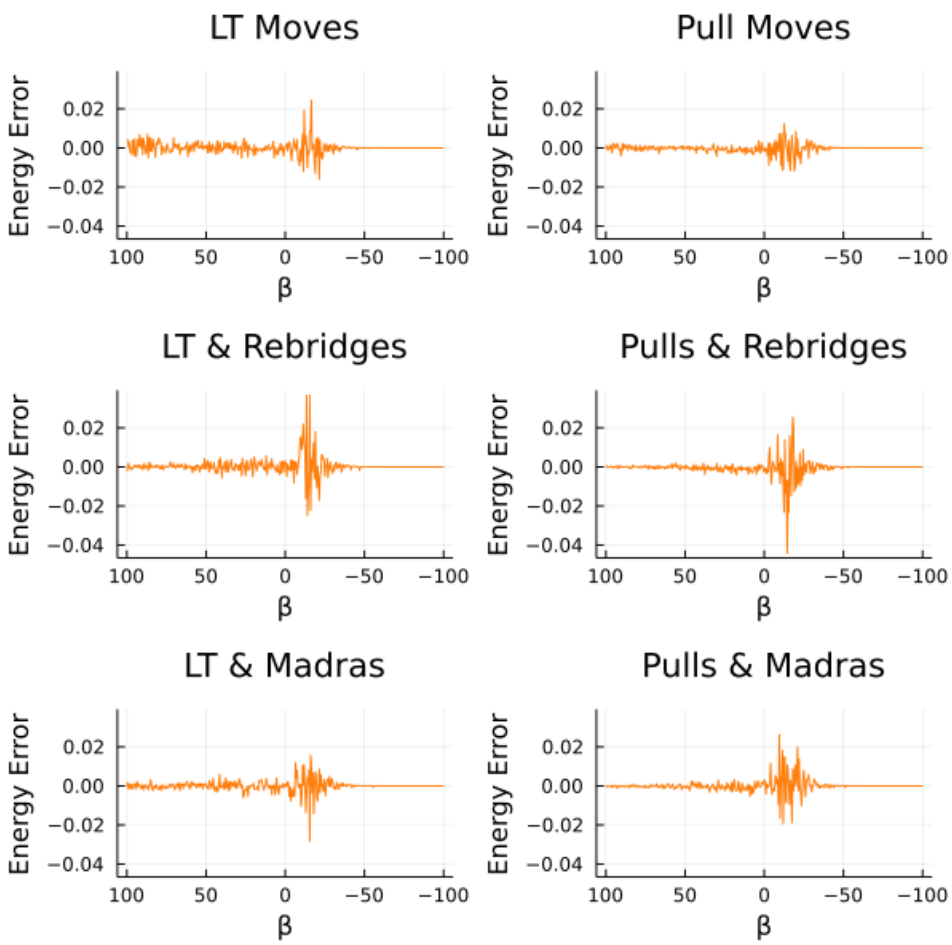


FIGURE 7. Absolute error between simulation-generated average energies and exact values; $N = 9$

low negative values of β, resulting in a higher variance from the exact value and a more "bumpy" curve in Fig. 6. This appears to be most prominent for combinations including rebridges, likely because the conditions required for a rebridge are unlikely to arise in high-temperature settings and the overall failure rate is higher as a result.

Of particular interest, however, is the region where β is large and positive. There is a noticable difference in this region, between the LT moves and pull moves, and to a lesser degree, with and without the rebridging or Madras moves. This matches expectations well; the latter three movesets were designed with protein folding in mind, and the goal in that field is generally to find the minimum-energy configuration. Naturally, the movesets created to solve this problem are quite a lot better at transitioning between these compact states with a minimal change in energy. In contrast, as Fig. 8 shows, the LT moves will frequently become "stuck" at one energy level for thousands of iterations at a time, causing a macroscopic change in the running average. This also occurs with pull moves, although seemingly only

at the lowest-energy configuration, resulting in a somewhat more accurate average. However, in both cases, these gulfs where no successful move is made disappear with the addition of the rebridging or Madras movesets.

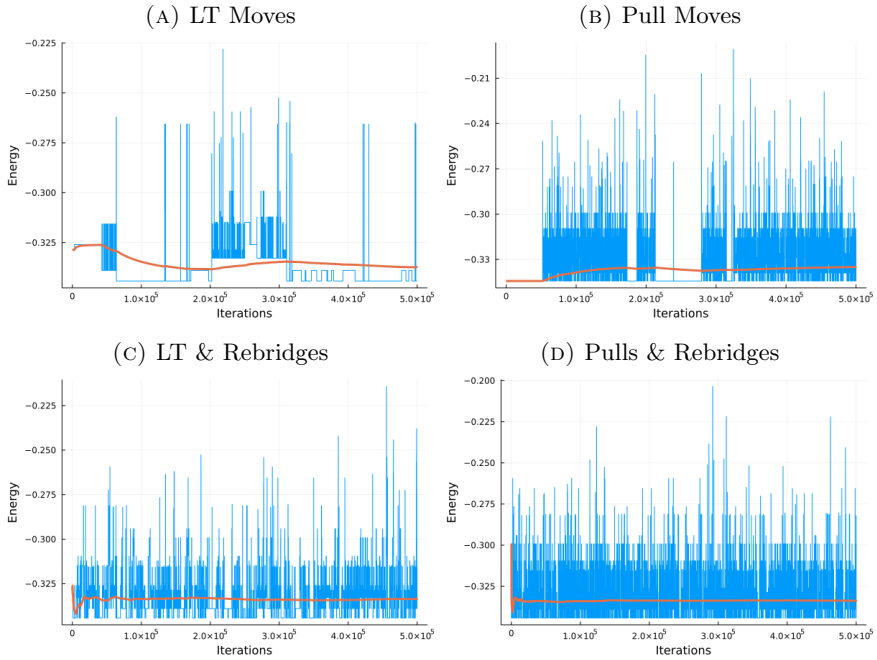


FIGURE 8. Plots of individual trials of 500,000 recorded iterations; $N = 9$, $\beta = 100$. The blue line shows the energy after each attempted move, while the red line is the running average. Trials with the Madras moves are not shown, as they had an effect similar to the rebridging moves.

4.2. **Optimal Moveset.** To assess the performance of each moveset, we turn our attention to larger values of N. For N from 10 to 100, with a step of 10, we ran a series of trials for each moveset for $-50 \leq \beta \leq 100$ and $\Delta\beta = 0.5$. Each trial starts with the final configuration from the previous trial[1], attempts 10,000 moves to burn-in to the new β value, and then records and averages the energy from a further 200,000 iterations. The final state of the previous trial is recycled for efficiency's sake, as it is expected to still be near the average when β is incremented a step. While the exact average energies have not been calculated for comparison, each moveset appears to still hold up at this scale, generating curves consistent with each other and the previous findings of Bělík et al.[1] The visible error across the values of β for each moveset also seem consistent with the previous, low-N results, although it appears a higher iteration count is needed across the board for the averages to better settle.

[1]A straight-vertical configuration is used at the beginning for $\beta = -50$

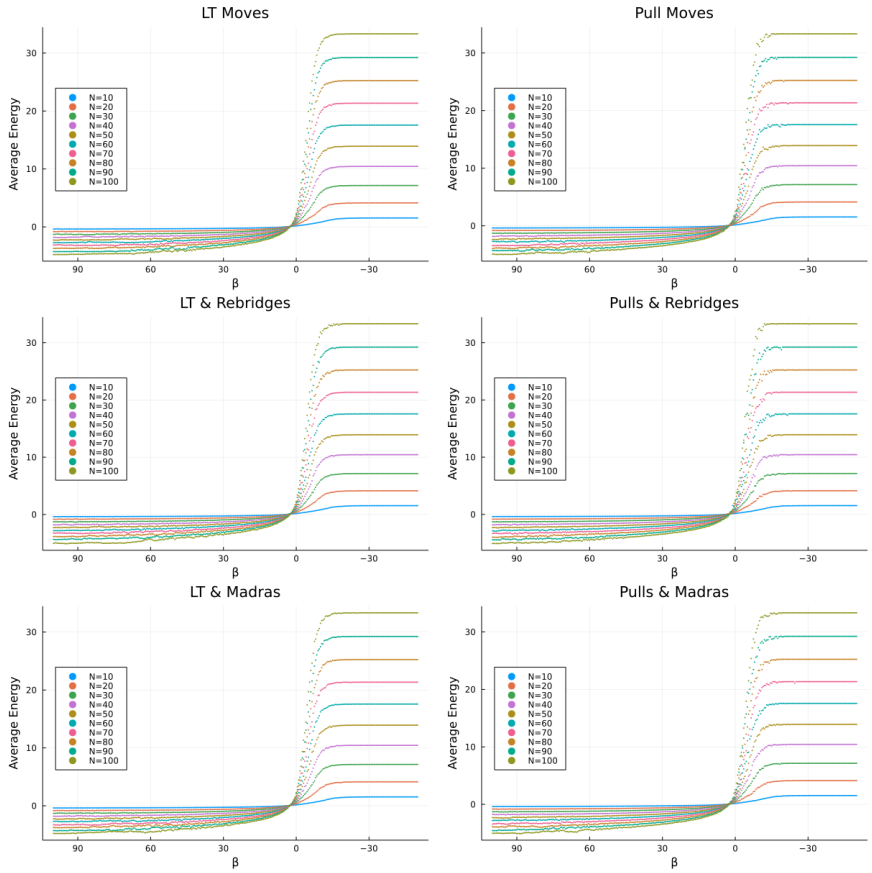


FIGURE 9. Simulation-generated average energies for $10 \leq N \leq 100$, $\Delta N = 10$, $-50 \leq \beta \leq 100$

It has, however, become apparent that the model performs best when two or more movesets are used in conjunction. The LT-alone and pull-alone combinations were dropped and two new combinations – pull-rebridge-Madras, and all four movesets together – were added in their place. We then attempted a more thorough trial with $N = 50$ and $-30 \leq \beta \leq 100$, upping the iteration counts to 50,000 burn-in and 1,000,000 energy-tracking. We also periodically calculate the standard deviation of energy for each trial, to better understand how close most moves were to the resulting average. As shown by Fig. 10, the variation for low-negative values of β ends up being largely consistent across all moveset combinations, while some still fail to fully settle in at the high-positive end. For this reason, we have concluded that the Pull and Rebridge movesets, used together, result in the most efficient and reliable average energy values overall.

4.3. **Minimisation.** While the maximum-energy configuration for each length follows a common structure and has a consistent formula, minima are far more irregular and difficult to confirm. To better understand the minima, it is first necessary

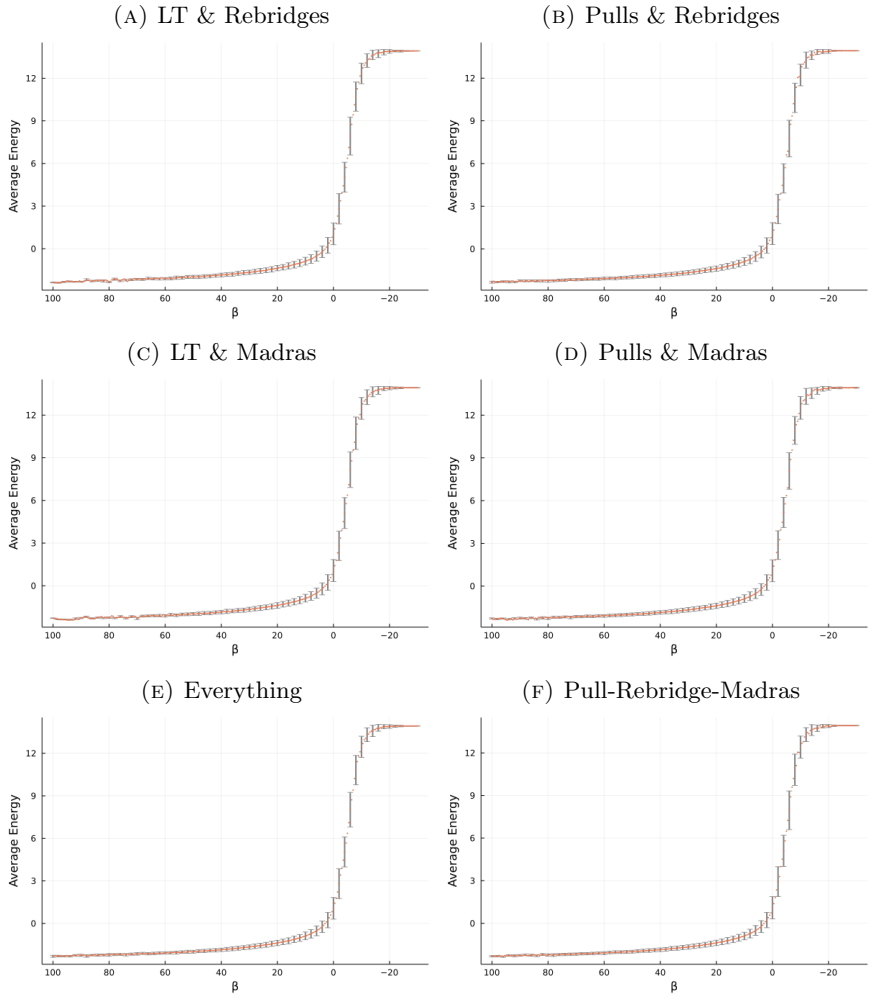


FIGURE 10. Average energy for $N = 50$, $-30 \leq \beta \leq 100$. Error bars mark standard deviation for every fifth trial.

to find them with reasonable certainty. To achieve this, we employed the Thermodynamic Simulated Annealing algorithm described by Vicente, Lanchares, and Hermida [6]. With this annealing schedule, we used each of the eight moveset combinations previously explored, attempting to find minima from $N = 3$ to $N = 90$. As N increased, it became clear that the LT, LT/rebridge, LT/Madras, pull, and later, pull/Madras and pull/rebridge combinations were less fit for the task and consistently unable to reach values as low as the pull-rebridge-Madras or "everything" combinations. As the underperforming combinations were removed, they were replaced with extra runs of the successful movesets so as to better confirm the found values. The result is that for every value of N, 6 to 8 attempts were made to locate a minimum, with some variety in the movesets used. This in turn helps to

validate each found value as a minimum, although there is still no guarantee that the true minimum was reached.

Using these techniques, we were able to create an extended list of potential minimum energy values for $N \leq 90$ (Appendix B). The results show a steady departure from the predictions of Bělík et al.; at the peak, a minimum found is more than 108% of the predicted value. However, results are inconclusive on whether the values grow linearly or at some other rate. We found two viable trends

$$E_{min} \sim a_1 N + c_1, \qquad E_{min} \sim a_2 N + b_2 N \log(N) + c_2,$$

where $a_1 \approx -0.0530462$, $c_1 \approx 0.180362$, $a_2 \approx -0.0336443 * i$, $b_2 \approx -0.00410912$ $c_2 \approx 0.0440087$, and $\log(n)$ is the natural logarithm function. While the linear option still fits well with $r^2 = 0.9994$, it is noticably higher than the data at the extremes, betraying a slight curve. For this reason the logarithmic option was attempted, and with a fit reported of $R^2 = 0.9999$, it matchs the growth better than any other attempted trend. This supports speculation that the minima, like maxima, grow logarithmically. While the search for greater-length minima could suggest a deviation from this trend, the near-perfect fit suggests that it will continue to hold.

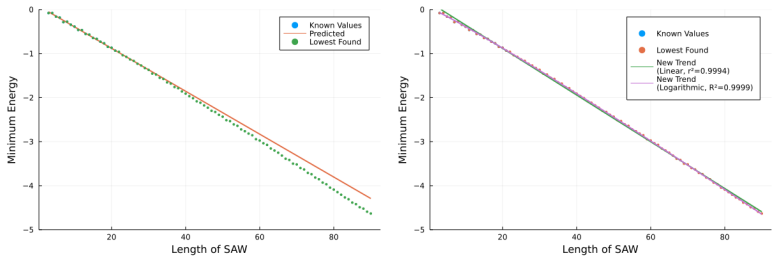


FIGURE 11. Minimum energy values for $3 \leq N \leq 90$, with trend lines. The left image shows the predictions of Bělík et al. based off of minima for $3 \leq N \leq 18$. The right shows best fit linear and $N \log(N)$ curves.

Examining the configurations themselves has also lead to some useful insights. First of all, all of the minima examined have endpoints within the same two dimensional unit square, supporting an observation by Bělík et al. [1]. It is not entirely clear why this may be, but related is the observation that nearly all minima for odd values of N exhibit bilateral symmetry. While this is not true of all odd values – $N = 25$ and $N = 29$, which were closely rechecked, lack full symmetry – this trend could be useful in finding further minima. One might be able to enact moves upon a SAW of length $\frac{N-1}{2}$, then reflect it across a plane and connect the two halves to check the energy. Beyond that symmetry, there are a number of structures and substructures found to be shared between and repeated within minima, visible in Fig. 14. This trend, which seems more pronounced for larger values of N, could also be utilised to find further minima. Unfortunately, there isn't necesarily a clear pattern to how one such minimum is related to the next or even follows the pattern at all. There is also the uncertainty of some configurations for being the true minima, as well as the unknown quantity of duplicate minima – non-congruent configurations that are each minima for the same length. It is already known that three trivial

duplicates exist, for $N = \{9, 15, 21\}$, but beyond these cases duplicates are difficult to construct or even prove or disprove the existence of. This area could benefit greatly from a more focused examination.

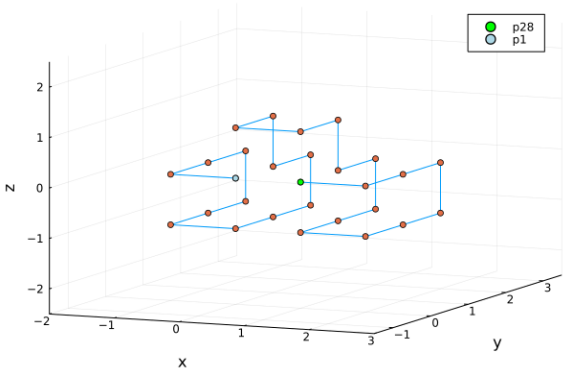


FIGURE 12. Minimum found for $N = 27$, exhibiting bilateral symmetry

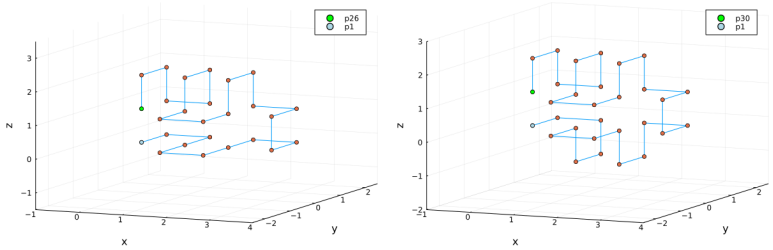


FIGURE 13. Minima found for $N = 25$ and $N = 29$. Note that each resembles a symmetrical configuration but with extra "wickets" replacing single segments and breaking the symmetry

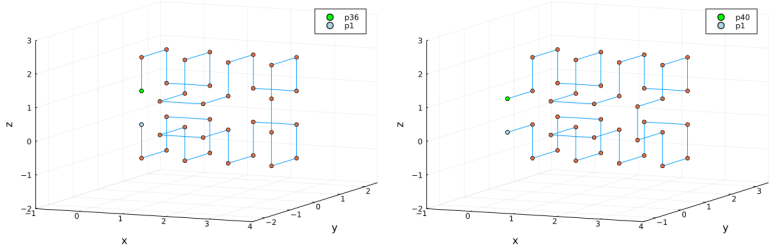


FIGURE 14. Minima found for $N = 35$ and $N = 39$, showing similar and repeating structures

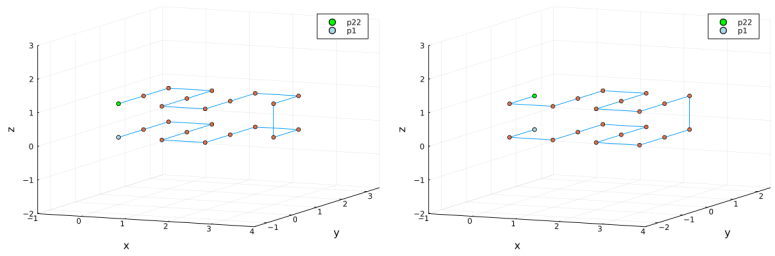


FIGURE 15. Two minima for $N = 21$

5. Conclusion

We have shown that pull moves, bond-rebridging moves, and the moveset decribed by Madras, Orlitsky, and Shepp are all viable movesets in this application. The combination of two or more of these movesets allows for more formidable and adaptive model. These three movesets in particular introduce a much greater freedom and stability for the low-energy configuration space, with a negligible loss of precision in the higher-energy range. Additionally, we've been able to use these movesets to find new minimum-energy configurations, up to $N = 90$. We now have an enhanced understanding of how minima are constructed and how they are distributed, and have even shown them to follow an $N \log N$ growth. It remains to be seen what will be found from further examination of low- and minimum- energy configurations, or from the finding of minima for still longer SAWs. These explorations can hopefully be made more efficient with the knowledge of how well these movesets all perform.

References

[1] Pavel Bělík et al. "Equilibrium Energy and Enrtopy of Vortex Filaments on a Cubic Lattice: A Localized Transformations Algorithm". In: (2021).

[2] J. M. Deutsch. "Long range moves for high density polymer simulations". In: *The Journal of Chemical Physics* 106.21 (1997), pp. 8849–8854. DOI: http://dx.doi.org/10.1063/1.473943.

[3] Neal Lesh, Michael Mitzenmacher, and Sue Whitesides. "A Complete and Effective Move Set for Simplified Protein Folding". In: (2003).

[4] N. Madras, A. Orlitsky, and L. A. Shepp. "Monte Carlo Generation of Self-Avoiding Walks with Fixed Endpoints and Fixed Length". In: *Journal of Statistical Physics* 58.1/2 (1990), pp. 159–183.

[5] Neal Madras and Alan D. Sokal. "The pivot algorithm: A highly efficient Monte Carlo method for the self-avoiding walk". In: *Journal of Statistical Physics* 50 (1988), pp. 109–186. DOI: https://doi.org/10.1007/BF01022990.

[6] Juan de Vicente, Juan Lanchares, and Román Hermida. "Placement by thermodynamic simulated annealing". In: *Physics Letters A* 317.5-6 (2003), pp. 415–423. DOI: https://doi.org/10.1016/j.physleta.2003.08.070.

[7] T. Wüst and D.P. Landau. "The HP model of protein folding: A challenging testing groundfor Wang–Landau sampling". In: *Computer Physics Communications* 179.1-3 (2008), pp. 124–127. DOI: https://doi.org/10.1016/j.cpc.2008.01.028.

[8] Thomas Wüst and David P. Landau. "Optimized Wang-Landau sampling of lattice polymers: Ground state search and folding thermodynamics of HP model proteins". In: *The Journal of Chemical Physics* 137.6 (2012), p. 064903. DOI: https://doi.org/10.1063/1.4742969.

Appendix A. Modified Proof of Pull Move Reversibility

Consider the moveset described by Lesh, Mitzenmacher, and Whitesides in the paper "A Complete and Effective Move Set for Simplified Protein Folding". For a SAW S of length n, we label the first vertex as w_0 and the last vertex as w_n[2]. If we define w_i and w'_i as the locations of a vertex on the SAW before and after a move, respecively, consider the "hook move", a kind of end-pull in which $|w'_n - w_{n-1}| = 1$ and, by extension, $|w'_{n-1} - w_n| = 1$. Also define a "square loop" as a set of consecutive vertices on a SAW $w_i, w_{i+1}, w_{i+2}, w_{i+3}$ such that $|w_i - w_{i+3}| = 1$. Note that any non-end-pull which displaces at least three vertices will create a new square loop and any move which does not propagate to an endpoint but displaces at least three vertices must collapse a square loop. These proofs will also use the term "steps" to refer to unit vectors aligned with the lattice grid, such that any points on the lattice w_i, w_j for which $|w_i - w_j| = 1$ are one step away from each other, or "adjacent".

THEOREM 1: All hook moves are non-reversible.

PROOF. After any hook move, two possible cases occur:

1. $w'_{n-3} \neq w_{n-3}$. If this is the case, $w'_{n-3} = w_{n-1}$ and, because $|w'_n - w_{n-1}| = 1$, a square loop is formed from $w'_{n-3}, w'_{n-2}, w'_{n-1}, w'_n$. To reverse this move, the last-moved vertex w'_i, where $i \leq n-3$, must be pulled back to w_i and the move propagated back along to the end of the SAW. However, once the vertex w'_{n-1} is returned to w_{n-1}, the move stops as $|w'_n - w_{n-1}| = 1$. Because w'_n is not returned to w_n, the resulting configuration is still distinct from the starting configuration.
2. $w'_{n-3} = w_{n-3}$. If this is the case, it must have been that $w_{n-3}, w_{n-2}, w_{n-1}, w_n$ formed a square loop which was collapsed by the hook move. To reverse this, the square loop must be recreated by pulling w'_{n-2} back to w_{n-2}. In turn w'_{n-1} moves back to w_{n-1}, but because $|w'_n - w_{n-1}| = 1$, the move stops there. Because w'_n is not returned to w_n, the resulting configuration is still distinct from the starting configuration.

It follows that for any hook move, no single pull move will revert the SAW back to its former configuration and therefore all hook moves are non-reversible.

THEOREM 2: All pull moves which are not hook moves are reversible.

[2]For the purposes of these proofs, I will consider only initial moves that propagate backwards along the SAW (possibly ending at w_0) and reverse moves which propagate in the alternate direction. Note that the moveset is symmetrical and that the proofs still apply when the vertices are labelled in reverse.

PROOF. Let M be a move upon S which is not a hook move, originates at w_a, and displaces every preceeding vertex up to and including w_b, where $0 \leq b \leq a \leq n$. There are many distinct categories M can fall into – three kinds of "special case" moves which only displace one or two vertices, and four kinds of larger-scale moves, depending on whether M started at w_n and whether M terminated before displacing w_0.

1. $b = a$. This occurs when w_{a+1} and w_{a-1} are two orthogonal steps away from each other, and w_a and w'_a are the two locations adjacent to both w_{a+1} and w_{a-1}. Because the only moved vertex was pulled from w_a, that location is open and the reverse move is trivial.
2. $b = a - 1$, $b \neq 0$ and $a \neq n$. This occurs when w_{a+1} and w_{b-1} are three steps away from each other, such that at least one step is orthogonal to the others, and the move consisted of pulling w_a and w_b to locations still adjacent to w_{a+1} and w_{b-1}, respectively. Because the only moved vertices were pulled from w_a and w_b to the distinct w'_a and w'_b, those locations are open and the reverse move is again trivial.
3. $b = 0$, $a = 1$. In this case, w_1 was pulled moving only w_0 with it. It must be that $|w'_0 - w_1| = 1$, and $|w_1 - w_0| = 1$, so it is trivial to reverse by making an end-pull and choosing the locations w_1 and w_0 to move w'_1 and w'_0, respectively.
4. $a - b > 1$, $b \neq 0$ and $a \neq n$. In this case, for the move to have terminated with w_b, it must be that $|w'_b - w_{b-1}| = 1$. $w'_b = w_{b+2}$, so $w_{b-1}, w_b, w_{b+1}, w_{b+2}$ is a square loop. w_b and w_{b+1} must also be left open by the first move, so one may reverse by pulling w'_b back to w_b. This propagates to w'_a, which goes back to w_a. However, $w'_{a-2} = w_a$ and $|w_a - w_{a+1}| = 1$, so $w'_{a-2}, w'_{a-1}, w'_a, w_{a+1}$ also forms a square loop. The reverse move then terminates after moving w'_a back to w_a, resulting in the original configuration. The reverse move cannot terminate before reaching w'_a, as that would require a square loop $w'_{i-2}, w'_{i-1}, w'_i, w'_{i+1}$ where $b + 2 < i < a$. If such a structure existed, there necessarily would have existed the square loop $w_i, w_{i+1}, w_{i+2}, w_{i+3}$, which would have terminated the move before it displaced w_b and therefore could not have existed.
5. $b = 0$, $1 < a < n$. In this case, the pull at w_a propagated all the way to the start of the chain, moving w_0 to $w'_0 = w_2$ and terminating. It is trivial to reverse by making an end-pull and choosing the locations w_1 and w_0 to move w'_1 and w'_0, respectively; the reverse move will necessarily end upon moving w'_a back to w_a.
6. $0 < b < n$, $a = n$. In this case an end-pull was made on w_n which propagated back to the square loop $w_{b-1}, w_b, w_{b+1}, w_{b+2}$ and collapsed it. Because M was not a hook move, $|w'_n - w_{n-1}| > 1$, and a pull on w'_b back to w_b will move w'_n back to w_n before terminating.
7. $b = 0, a = n$. In this case there exist no square loops in the initial configuration, and because M is not a hook move, there are no pairs of vertices $|w'_i - w_{i-1}| = 1$. As such, an end-pull from w'_1 and w'_0 to w_1 and w_0, respectively, will return all vertices up to and including w_n to their original position.

Above is listed every possible case where M is not a hook move, and in each case it shows that there does exist at least one single pull move to reverse M and

obtain the original configuration. Therefore, the set of pull moves, excluding hook moves, is always reversible.

Appendix B. Table of Newfound Minima

N	$E_{\min}$ predicted	$E_{\min}$ found	% of predicted	times found
3	-0.051498083	-0.079577472	1.545%	7
4	-0.100232759	-0.079577472	0.794%	7
5	-0.148967435	-0.159154943	1.068%	7
6	-0.197702111	-0.182462645	0.923%	7
7	-0.246436787	-0.285347818	1.158%	7
8	-0.295171463	-0.275022123	0.932%	7
9	-0.343906139	-0.344273900	1.001%	7
10	-0.392640815	-0.387531410	0.987%	7
11	-0.441375491	-0.462065557	1.047%	7
12	-0.490110167	-0.467078630	0.953%	7
13	-0.538844843	-0.552979972	1.026%	7
14	-0.587579519	-0.568683986	0.968%	7
15	-0.636314195	-0.644807616	1.013%	6
16	-0.685048871	-0.666143929	0.972%	6
17	-0.733783547	-0.730433740	0.995%	7
18	-0.782518223	-0.766861383	0.980%	6
19	-0.831252899	-0.846227657	1.018%	7
20	-0.879987575	-0.867780588	0.986%	6
21	-0.928722251	-0.932714628	1.004%	5
22	-0.977456927	-0.959646702	0.982%	7
23	-1.026191603	-1.037495274	1.011%	5
24	-1.074926279	-1.074012935	0.999%	5
25	-1.123660955	-1.132796476	1.008%	7
26	-1.172395631	-1.166089163	0.995%	6
27	-1.221130307	-1.231197530	1.008%	6
28	-1.269864982	-1.271218455	1.001%	6
29	-1.318599658	-1.329584796	1.008%	7
30	-1.367334334	-1.369574239	1.002%	4
31	-1.416069010	-1.455336167	1.028%	3
32	-1.464803686	-1.475574830	1.007%	3
33	-1.513538362	-1.550603167	1.024%	3
34	-1.562273038	-1.576860519	1.009%	4
35	-1.611007714	-1.654298719	1.027%	3
36	-1.659742390	-1.682140307	1.013%	4
37	-1.708477066	-1.760069956	1.030%	7
38	-1.757211742	-1.787300364	1.017%	7
39	-1.805946418	-1.859662168	1.030%	3
40	-1.854681094	-1.909390372	1.029%	6
41	-1.903415770	-1.966829156	1.033%	2
42	-1.952150446	-2.013764256	1.032%	4

N	$E_{\min}$ predicted	$E_{\min}$ found	% of predicted	times found
43	-2.000885122	-2.084571314	1.042%	4
44	-2.049619798	-2.116372906	1.033%	4
45	-2.098354474	-2.176894874	1.037%	3
46	-2.147089150	-2.229137124	1.038%	5
47	-2.195823826	-2.299049013	1.047%	3
48	-2.244558502	-2.326301351	1.036%	2
49	-2.293293178	-2.391019098	1.043%	4
50	-2.342027854	-2.432964747	1.039%	6
51	-2.390762530	-2.513125703	1.051%	4
52	-2.439497206	-2.533875575	1.039%	1
53	-2.488231882	-2.608068265	1.048%	4
54	-2.536966558	-2.640364014	1.041%	2
55	-2.585701234	-2.722068683	1.053%	4
56	-2.634435910	-2.763284951	1.049%	5
57	-2.683170586	-2.816661217	1.050%	2
58	-2.731905262	-2.854915219	1.045%	6
59	-2.780639938	-2.944441781	1.059%	4
60	-2.829374613	-2.974806268	1.051%	6
61	-2.878109289	-3.037352114	1.055%	5
62	-2.926843965	-3.073585680	1.050%	5
63	-2.975578641	-3.155357636	1.060%	6
64	-3.024313317	-3.201457333	1.059%	6
65	-3.073047993	-3.251241733	1.058%	1
66	-3.121782669	-3.315632696	1.062%	5
67	-3.170517345	-3.388322568	1.069%	6
68	-3.219252021	-3.416254436	1.061%	6
69	-3.267986697	-3.500258989	1.071%	6
70	-3.316721373	-3.517352133	1.060%	4
71	-3.365456049	-3.599391745	1.070%	4
72	-3.414190725	-3.632155798	1.064%	5
73	-3.462925401	-3.709473591	1.071%	6
74	-3.511660077	-3.739430833	1.065%	4
75	-3.560394753	-3.815585195	1.072%	2
76	-3.609129429	-3.853881048	1.068%	4
77	-3.657864105	-3.930081745	1.074%	1
78	-3.706598781	-3.967677486	1.070%	2
79	-3.755333457	-4.044563525	1.077%	2
80	-3.804068133	-4.088979759	1.075%	3
81	-3.852802809	-4.146215704	1.076%	2
82	-3.901537485	-4.205374510	1.078%	6
83	-3.950272161	-4.266635286	1.080%	4
84	-3.999006837	-4.308439419	1.077%	6
85	-4.047741513	-4.384310101	1.083%	5
86	-4.096476189	-4.419576316	1.079%	5
87	-4.145210865	-4.489772412	1.083%	6
88	-4.193945541	-4.521148388	1.078%	6
89	-4.242680217	-4.591070327	1.082%	2

N	$E_{\min}$ predicted	$E_{\min}$ found	% of predicted	times found
90	-4.291414893	-4.632405948	1.079%	4

Appendix C. Notes on Maintaining the Detailed Balance

Our approach requires a general balance $P(B) = \sum_S P(S)P(S \to B)$, but we will instead be enforcing the stronger detailed balance $P(A)P(A \to B) = P(B)P(B \to A)$. This is because the detailed balance is much easier to keep track of and implement. For this purpose we use the Metropolis equation $A(x', x) = \min\left(1, \frac{P(x')}{P(x)}\frac{g(x|x')}{g(x'|x)}\right)$.

As implemented by the code, the proposal probabilities for pull moves are not always equivelent and so $\frac{g(x|x')}{g(x'|x)}$ cannot be assumed to be 1. This is because the code uses the following algorithm to make a move:

1. Choose a vertex w_i on the SAW. $(\frac{1}{n})$
2. (a) If the vertex an endpoint w_0, pick a direction d_1 other than the step s_0. $(\frac{1}{5})$
   (i) If d_1 is equal to $-s_0$, pick another direction $d_2 \neq -d_1$ $(\frac{1}{5})$
   (ii) If d_1 is not equal to the direction $-s_0$ pick another direction $d_2 \neq -d_1, d_2 \neq s_0$. $(\frac{1}{4})$
   Make an end-pull with the given parameters.

   (b) If the vertex is not an endpoint, pick a direction ± 1 along the SAW. $(\frac{1}{2})$
   Then pick a unit vector which is orthogonal to either s_i or s_{i-1}. $(\frac{1}{4})$
   Make a pull with the given parameters.

3. Attempt the move. Check for self-intersection
4. Determine if the move should be accepted.

The probability that a given pull move is proposed therefore ranges from $\frac{1}{25n}$ and $\frac{1}{20n}$ for end-pulls to $\frac{1}{8n}$ for interior moves[3]. There are therefore moves which have a different proposal probability than their reverses, and $\frac{g(x|x')}{g(x'|x)}$ must be explicitly calculated. This is made easier by the hyposthesis that every move has exactly one reverse, as returning to a congruent but not equivelent state is not considered a reversal. That last condition is largely for convenience and may be discarded if it is deemed necessary (or convenient) to do so.

[3]While interior moves for which w_i is the only point displaced have a $\frac{1}{4n}$ chance of being proposed, there is always an equivelent chance of the reverse being proposed, so they do not disrupt the detailed balance.

Appendix D. Notes on Implementation – Max Figura

I wrote the code for this simulation in Julia, using the `plots` package for visualisation purposes. All code was written by myself; no segment was copied directly from an external source. I make no claim to the speed or efficiency of the code, particularly because I had little-to-no experience with this particular language before undertaking the project. Additionally, some of the source materials for each of the movesets and algorithms lack the detail necessary to be consistently implemented in practice. To help amend this issue, and in the interest of transparency and reproducibility, I would like to share the following specifications on how the code I've created works.

Basic Functionalities:

- The SAW itself is kept track of by three collections:
  - An array of integers, valued 1-6, each representing a cardinal direction, so as to show the "steps" from each point to the next.
  - An array of three-element tuples, serving as the ordered list of coordinates visited by the SAW.
  - A `Set` of three-element tuples, with all the same values as the coordinate array, used to determine if a given point is on the SAW.

  The coordinate array is used for a majority of calculations, but the direction array is instrumental for the energy calculation (determining if two segments are orthogonal) as well as determining in what direction a move can be made. Functions were also written to generate the coordinate array from the direction array (recalibrating the starting point of the SAW at the origin) as well as the reverse, so that a move could mutate only one structure and then update the other accordingly.
- In keeping with the conventions of Julia, all data structures are 1-indexed and therefore here the start of the SAW is equivelent to "point 1".
- A simple distance-calculating function is used for the energy calculation, as well as in determining if a given pair of points is adjacent.
- Two versions of the energy-calculating function were written:
  - The straightforward approach. For each step in the SAW, iterate over every other step, determining if the segments are parllel/antiparallel or skew. If the former, calculate midpont distance and contribute to the sum as appropriate.
  - The sorting approach. Iterate over the steps of the SAW, sorting the midpoint of each according to its direction. For each point in a direction category, iterate over the other points in that direction as well as those in the reverse; find the distance and contribute to the sum as appropriate.

  Theoretically, the sorting approach should take about a third of the computation time as the straightforward approach, depending on the distribution of direction within the SAW. When comparing individual calls on randomly-generated SAWs, this seemed to be the case. However, when tested with trial runs of the simulation itself, the straightforward approach seemed to perform better; while the sorting approach was naturally slower for the straight configurations resulting from large negative values of β, the lead

it gained on the positive end was significantly smaller. It is possible that this discrepancy is due to using shorter-length SAWs for benchmarking, or errors in the benchmarking process itself. At time of writing, the straightforward approach has been used for a majority of calculations.

LT Moves:

- The function for the random reconstruction takes a single index and an array of randomised replacement directions; after splicing the replacement into the direction array starting at the index, that entire latter section of the coordinate array is recreated using the direction array. This has effect of simultaneously reconstructing the changed section and translating one end to match. The recreation is performed one step at a time so that self-intersection can be checked one point at a time; if no intersection is detected, the move was successful.
- While the (cyclic) permutative reconstruction could be achieved by feeding a particular direction array into the random reconstruction function, it was decided that it was inefficient to recalculate the position of the unchanged section as it will necessarily be unmoved. Instead, the function takes the start and end index as well as a number of times to cycle the directions between. This section of the direction array is spliced out, cycled, and respliced, and the appropriate section of the coordinate array is reconstructed similarly to in the random reconstrucntion.
- An individual function was written for the purpose of calculating the exponential distribution of move lengths. The distribution takes the form of an array of cumulative probabilities, such that a random value between 0 and 1 can be generated and checked against each in successtion. The distributions are generated once at the beginning of a trial, such that they can be passed as arguments every time a move is made.
- To make a random LT move, it is first randomly decided which of the reconstructions to perform, then in either case a move length is decided according to the distribution. After choosing an index in accordance with the move length, either a number of cycles[4] (in the case of the permutative reconstruction) or an appropriate-length array of random directions (in the case of the random reconstruction) is generated. The moves are attempted, energy calculations are performed, and the move is accepted or rejected as a result.

Pull Moves:

- The pull move itself is divided into two seperate functions:
  - The regular pull move, which takes a point along the SAW to be pulled, a displacement vector for the point to be relocated, and a binary value for direction along the SAW for the move to be propagated. After checking for self-intersection, it determines the "bypoint" and checks if the move only relocates the single given vertex. It then inserts the destination and by points into the coordinate array, determines where the move ends, and finds and removes from the array the points

[4]Not including the identity permutation

that are no longer on the SAW. In this way those vertices unaffected by the move are kept at the same index, while the rest are shifted appropriately.
  - The end move, which takes a point along the SAW to be pulled, a displacement vector for the bypoint, and a displacement vector for the destination point. once it checks for self-intersection, it plays out similarly to the regular pull moves which displace multiple vertices.
- In order to attempt a random pull move another function was created to randomly choose a point, binary direction, and displacement vector (or point and two displacement vectors, in the case of end moves). The validity of the move is then checked by applying the appropriate pull function to a test array, a copy of the current configuration. If that is successful, it will calculate the energy of the current and proposed configurations and generate a final random value to determine if the move should be accepted (as explained above, this probability is further weighted by the imbalanced chance of proposing a given move). The SAW is then mutated to match the test version, if appropriate, and then the function returns whether successful or not.

Rebridging Moves

- In its most basic form, the rebridging move is represented by splicing some section of the SAW (the section "between" the points to be connected) out of the coordinate array, transforming it in some manner, and splicing it back in at a potentially different index. In this way the direction along the SAW from the start to the endpoint is kept intact, even if subsections are reversed.
- With this implementation, a move is defined by a point and its successor along the SAW, and a direction for each to be rebonded to. Because of this, single-step moves are each achievable in two different ways, although the second step of a double-step move can only be picked one way, as the chosen point must be on the "loop" section.
- We perform a random move by picking an index between 0 and N, where these two extremes correspond to an end move and all other values correspond to a rebridging "starting" at that point. Note that there are therefore two distinct kinds of moves starting at point 1.
- Due to the compound nature of the two-step move, validity checks cannot all be performed after choosing one distinct proposal. Instead, if a mid-SAW move is chosen, after an orthogonal direction is picked, a check is performed on whether a rebridging is possible in the direction, where the connection would be made to, and whether the move will necessarily be one or two steps.
  - Single-step: In determining the type of move required, we have necessarily verified that a single-step rebridge is possible and therefore can immediately splice the appropriate section of the coordinate array, reverse it, and replace it.
  - Double-step: Another point and orthogonal direction are chosen, ensuring that the second point is on the loop section and therefore potentially reconnecting it to the rest of the SAW. A check is made if

the move would return the SAW to its starting configuration (If so, it is tracked as a successful move but the configuration is unchanged), then it is checked whether the second move could actually reconnect the loop to the rest of the SAW. While there are still multiple remaining cases, it was found that the next course of action could be determined soley by whether the current directions at the points to be connected are parallel or anti-parallel. The loop is "rotated" (and perhaps reversed) by cycling points in the array until the desired point is at the start, then it is replaced at the appropriate location back into the coordinate array.

- End moves are perhaps the simplest to implement; once the point on the SAW in the appropriate direction from the endpoint has been located (if found to exist), one may splice, reverse, and replace the coordinate list between that point and the endpoint.

Madras Moves:

- Unlike the previous movesets, the moves described by Madras, Orlitsky, and Shepp all behave rather similarly; given a sub-section of the SAW and possibly a few other parameters, they will each effectively replace that section with a transformed version of itself. In each case, this is done by splicing the transformed points out of the coordinate array, using them to create a replacement segment, checking that the replacement will not result in intersection, and re-splicing it back in.
- In the "reflection" move, each point in the replacement is constructed one dimension at a time. Starting with the vector `[0,0,0]`, the two given dimensions are assigned according to the formula described in the paper, then the remaining value(s) are filled in according to the modified formula given by Bělík. It is then checked if the given point is already on the remaining SAW, and so on until the entire replacement is constructed.
- The "interchange" plays out similarly, finding the appropriate displacement vector for the point's step direction, reassigning the values according to the interchange, and using that vector to determine the next point of the replacement segment.
- The "inversion" is actually achieved with the reflection function, as it was found that inputting the same dimension for both parameters (and a slope of 1) results in the same action. While a function was written to instead reverse a section of the direction array, it was determined to be generally slower than re-using the reflection function.
- In choosing a random move, two distinct points are chosen, then a move type. If not an inverse, a pair of dimensions and a slope are also chosen. This is enough to define a move, so the move is then attempted and checked for energy. As each move is its own reverse, and the specific configuration of the SAW does not affect the chances of a given move being proposed, the proposal probablility of a given move matches the proposal probability of the reverse and no modifications need to be made to preserve the detailed balance.

Thermodynamic Simulated Annealing:

- Many variants of the TSA algorithm were implemented, often due to errors or misunderstandings in the correct formulation, but some showed surprising levels of success. The final version, however, was deemed to perform the best and only differed from the pseudocode by Vicente, Lanchares, and Hermida in two key ways. For every rejected move, the algorithm skips on to the next iteration without updating entropy (regardless if the move would have increased energy or not), and for any iteration where the temperture T_k is calculated to be less than or equal to 0, or equal to ∞, it sets $T_k = T_{k-1}$ rather than $T_k = T_0$.
- The background math in the original paper suggests that TSA holds valid even when the "window" over which total energy and entropy change are calculated is smaller than the full trial. For this reason, multiple variations were attempted with a "moving window" with which to calculate the next temperature value. However, this caused the temperture value to behave erratically, becoming more stable when larger windows were used. The logical conclusion was that it was most stable when the window was larger than the full trial itself, and the moving window approach was discarded.
- In the final form of the algorithm, a stopping value of $T_{end} = 0.01$ is used, although an additional hard limit on the number of iterations is imposed to prevent trials from running for too long as well as watching for unexpected behavior. A runtime/quality parameter of $k_A \approx 33$ was intitally used for single-digit SAW lengths but is slowly increased as trials are run with longer lengths.