Continuous Monte-Carlo Protein Folding

Bhareth Kachroo

December 20, 2017

1 Introduction

Simulated protein folding is a minimization problem. A protein is such a complex, heterogeneous molecule that a direct simulation of Schrodinger or even Maxwell's equations is not currently feasible for realistic chain sizes (300 amino acids). Instead we assume thermal perturbation will eventually bring the protein to the deepest well in the energy landscape. If there were many minima of comparable sizes this would be useless, but evolution designs proteins that are robust to small changes in sequence, with a funneled energy landscape and guided folding process. As a result, real proteins fold much faster than we would expect for random thermal sampling [2].

The goal here is to capture some of the essential physics associated with this energy landscape using a simplified model, based on Giordani and Nakanishi's (GN) approach. [1] There are several factors influencing the landscape: the 20 types of amino acid interacting with each other, water molecules interacting with the amino acids, molecular chaperones and foldases that assist in folding. The GN model attempts to reproduce only the acid-acid interactions. This new model tries to improve on some of GN's deficiencies.

The most commonly used first principles approach is Monte-Carlo simulation, which is also GN's approach. The most successful approach is actually homology modelling, using prior data on protein structure to look for patterns, but we're interested in the physics.

2 Model

2.1 GN Model

GN represent each amino acid as a point and the protein as a chain on a 2D grid, with the minimum distance equal to the bond length. The energy landscape is traverse by randomly moving an acid position to a nearest-neighbour such that the bond lengths remain fixed, approximating the strong energy associated with covalent bonds as opposed to other acid-acid interactions. The protein begins as a straight line, and follows the Metropolis algorithm.

The energy is calculated based on a randomized interaction between each type of acid at the start of the simulation. The interaction is only for adjacent acids that are not next to each other on the chain.

This model has so many constraints it is unsuitable for a Monte Carlo simulation; it takes 25 steps to make the first move because only the endpoints can move at the beginning. This doesn't get much better: after 249 steps the result is only Fig. 1.

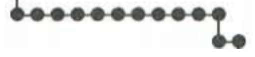

Figure 1: Folding after 249 time steps [1]

This approach is extremely inefficient. It is also unstable, as shown by tests of the same protein folded repeatedly (with identical interactions) finding significantly different energy levels.

2.2 Continuous Model

The model proposed here is mainly addressing the inefficiency of the GN model. Proteins are represented the same way, but in continuous 3D space. Varying the positions of the acids requires accounting for covalent bond energies and vibrations along the chain, so the variable to change is instead absolute angle in spherical coordinates. The bond length is preserved and now any angle can move, hugely reducing the number of wasted steps. If instead relative angle had been used, one section of the protein would rotate against the other, which would make progress in the later stages when the protein is mostly folded very slow. Instead one acid rotates and drags the rest of the protein over slightly, similar to the effect of pulling on coupled springs. See Figs. 2 and 3 for what this looks like.

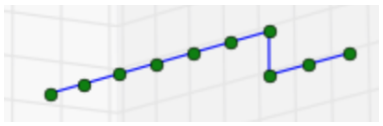

Figure 2: Second Step

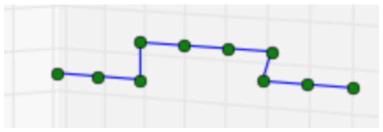

Figure 3: Third Step

The energy calculation is also more realistic. Checking each unique acid pair for adjacency is $O(n^2)$, so with the same asymptotic behaviour we can calculate each unique pair distance and use a radial potential to calculate the interaction energy. The potential used is two competing exponentials, as an approximation of the Lennard-Jones potential:

$$U(r) = 8e^{-\frac{r}{0.2}} - e^{-\frac{r}{5}} \tag{1}$$

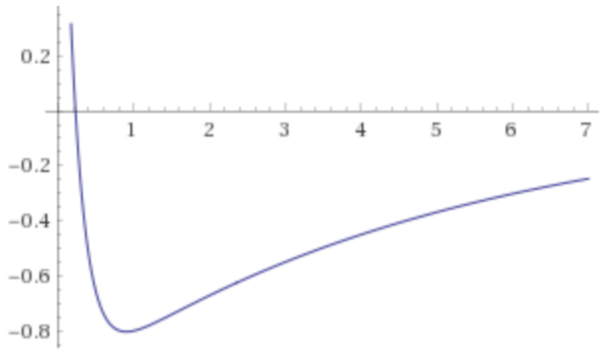

Figure 4: Plot of Potential

The potential is shown in Fig. 4. The idea is to encourage acids to be around the range of the bottom of the well, which is close to the bond length (one). This potential can be randomized to simulate different types of acids by changing the relative size of the coefficients and exponential parameters, but the repulse term should always have a larger coefficient and smaller exponential parameter. The effect would be to change the depth of the well, its position, or the intensity of the near-field repulsion.

There is a restriction on how sharply the protein can bend at the joint of an amino acid, beyond what the potential provides, as the real amino acids are not located entirely at one point. This is approximated by restricting the relative angle between a triad of three adjacent acids to be greater than $\frac{\pi}{2}$ radians.

3 Computational Method

Most of the new code is for geometry. Calculation of the potential requires absolute positions of the acids, called `positions()`. Since the first acid never moves, we start with that as the origin. Every successive absolute pair of angles is available. Transform the spherical coordinates with radius one to cartesian coordinates, and add these to the coordinates of the previous acid to get absolute coordinates for the next acid.

The restriction on relative angle is maintained by a mini-Monte-Carlo method: randomly shift the angles until one that isn't too acute is found (done by `try_angles()`). This rarely takes more than three tries. The relative angle is calculated by using the central acid as the origin and finding the great circle distance between the two positions on the sphere, given latitude and longitude (by `relative_angle()`). This gives angle:

$$A = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos(\Delta\theta)) \tag{2}$$

where ϕ is the longitude and θ is the latitude.

The Metropolis algorithm (in `metropolis()`) picks a random index, calls `try_angles()` to find a pair of new random angles, and recalculates the positions and total energy, using the Boltzmann factor for probability if the new energy is higher than the previous one.

Accuracy is not well-defined, as we are not approximating a known mathematical structure, but trying to elucidate some properties of a physical process. The model is so simplified that the two cannot be directly compared beyond gross properties like stability.

In a randomized simulation the error is typically just more randomness; here the major source of numerical error is the energy calculation, but the magnitude is still tiny even for realistic sized chains of n=300. This floating-point error only becomes significant for small magnitudes or an extremely large number of operations, neither of which apply. The end result of any error is simply something similar to operating with a slightly higher Boltzmann temperature.

The most computationally demanding step is the energy calculation, which calculates distance and an energy exponential $n^2/2$ times, where n is the number of amino acids. All other steps, like the $O(n)$ position calculation, are asympototically irrelevant. Then the time for a single Monte-Carlo step is $O(n^2)$ and the total time is $O(mn^2)$ where m is the number of steps. The memory requirement is just $O(n)$ to store the angles. Given the true protein-folding problem is NP-hard, those are pretty good efficiencies.

3.1 Testing

Direct tests are visual: are the acids too close together, do the angles change, are angles too acute. The method passes visual tests. Since the simulation is randomized, other tests are of component functions. The `metropolis()` function is straightforward and was tested in Lab 11, by checking the proportion of accepted moves vs. the energy difference. Relative angle is tested by trying it on some example angles and calculating by hand. Positions is tested by trying some simple starting angles and calculating the positions by hand. `try_angles()` is tested by printing out the random angles it chooses and seeing which it returns, and whether it is in the appropriate range.

4 Results

As seen in Figs. 5 - 8 the protein seems to become sufficiently complex by step 40 that after that it is impossible to distinguish which step it is by sight. Fig. 5 and 6 are from the same run, while the other two are both separate runs, as contrast.

The energy varies with time as a scale-free power law, as seen in Figs. 9 and 10. Compare this to Fig. 11 and the difference is striking. The additional degrees of freedom clearly improve the speed of convergence, even at low T=1.

The algorithm is sensitive to temperature differences, as seen in Fig. 12, which is far more chaotic than T=1. In addition the higher temperatures result in a more strung-out chain as in Fig. 13.

The behaviour of the best yet energy is fascinating, in Figs. 14 and 15. They also have scale-free power law curves, but the changes are sudden breakthroughs rather than steady improvements. The distance between breakthroughs also increases exponentially, and their size decreases. The total energy behaviour

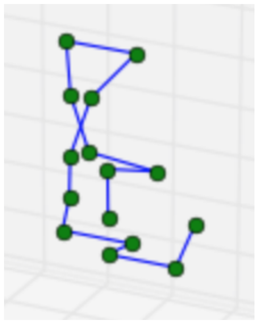
Figure 5: Step 30

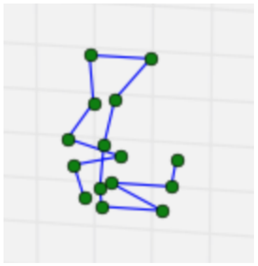
Figure 6: Step 40

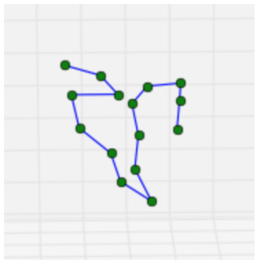
Figure 7: Step 100

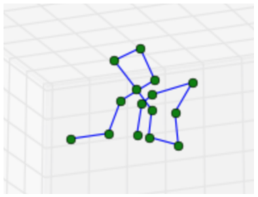
Figure 8: Step 200

shows the system falling into deeper and deeper wells, and then climbing out. The early section shows funnel-like energy behaviour, with every run quickly falling toward a folded rather than extended structure. The energy range of minima after 50 steps is small enough (2, compared to a 33 difference between these states and unfolded) that these could be called substates at a general minimum.

The very long-range behaviour shows a stabilization over time. The limit for this energy function at a constant temperature seems to be around -65. As in Figs. 16, 17 the higher temperatures stabilize around a higher average energy, with a larger deviation. The density of the wells appears to stabilize as well, suggesting a periodic aspect to the modulation, which deserves spectral analysis.

There is also improved stability for repeated runs as compared to the GN model in Fig. 18. Fig. 19 shows energies never quite settling down, but clearly occupying the same range, as opposed to different levels in Fig. 18.

Figs. 20 to 23 shows the two structures at step 5000 from Fig. 19. While they are not identical, they have clear structural similarities as compared to the other figures of protein structures above. The structural stability of the algorithm and energy landscape suggest this approach is accurate in capturing the general properties of real protein folding.

5 Conclusion

The approach discussed here is a much better representation of protein folding than the original GN model, displaying funneling and structural stability on repetition. While the energy calculation is more complicated, sensible degrees of freedom reduce the number of Monte-Carlo steps required (from $20*10^5$ to 100) so greatly that it runs much faster.

Next steps would be randomizing the potential energy variables, examining the effects of simulated annealing, and seeing if any typical secondary protein structures can be reproduced like alpha helices.

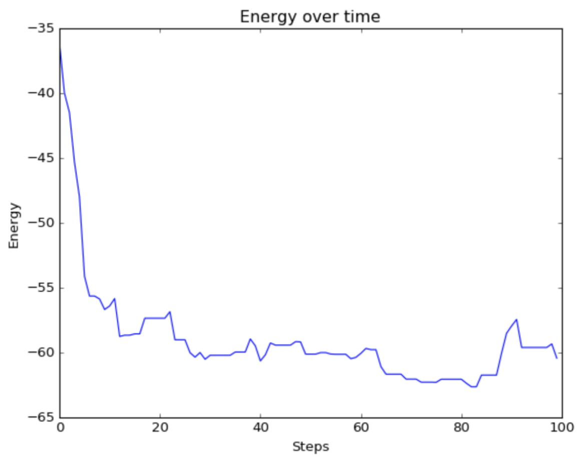

Figure 9: Energy v. Time for 100 steps

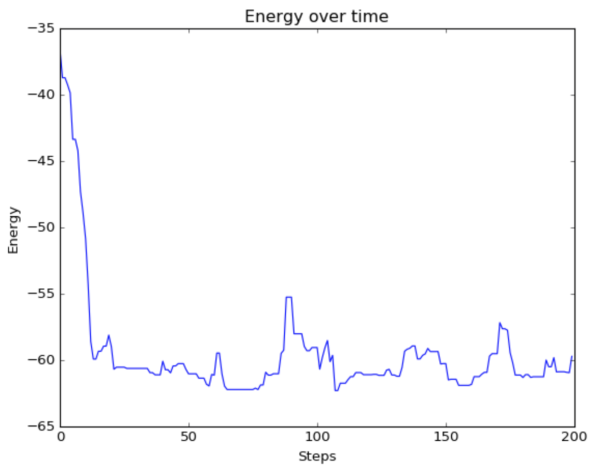

Figure 10: Energy v. Time for 200 steps

References

[1] Giordano, N. and Nakanishi, H. (2006). *Computational physics*. New Delhi: Dorling Kindersley.

[2] Chen, M., Lin, X., Zheng, W., Onuchic, J. and Wolynes, P. (2016). *Protein Folding and Structure Prediction from the Ground Up: The Atomistic Associative Memory, Water Mediated, Structure and Energy Model*. The Journal of Physical Chemistry B, 120(33), pp.8557-8565.

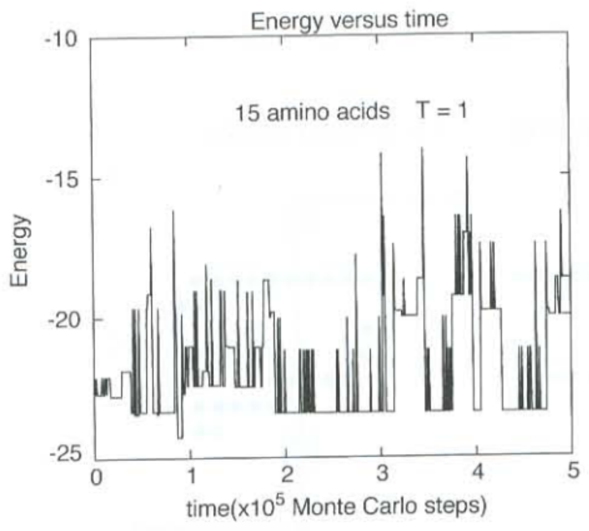

Figure 11: GN's Energy vs. time [1]

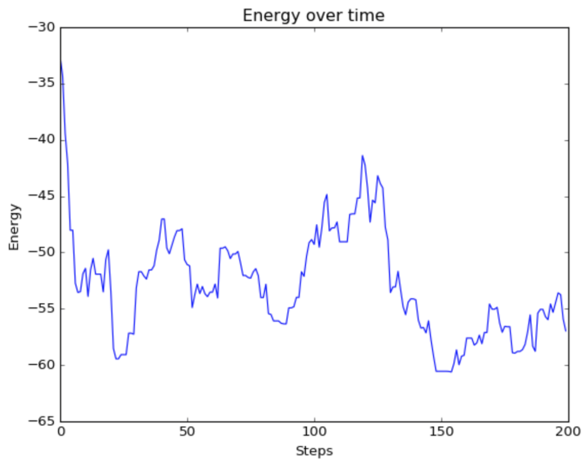

Figure 12: Energy vs. Time for T=7

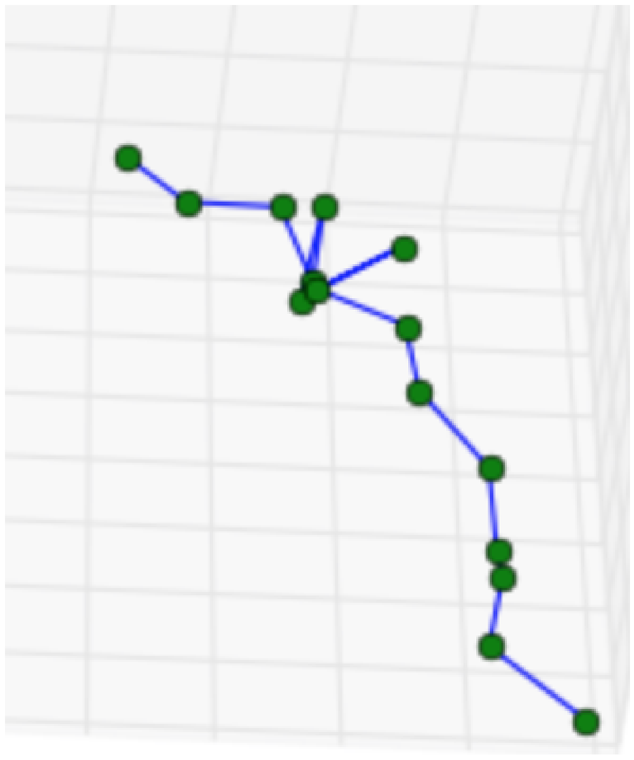

Figure 13: Chain at step 100 for T=10

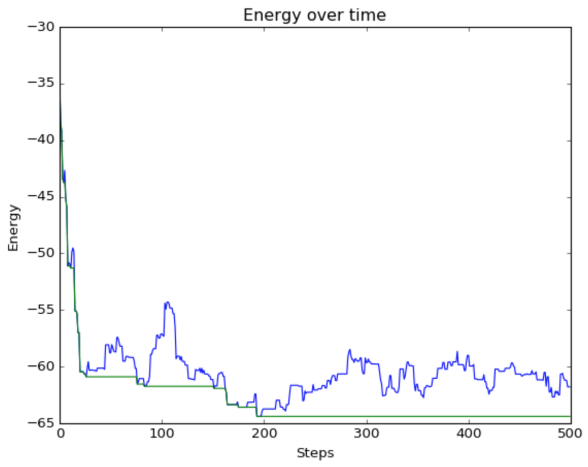

Figure 14: Best Energy v. Time

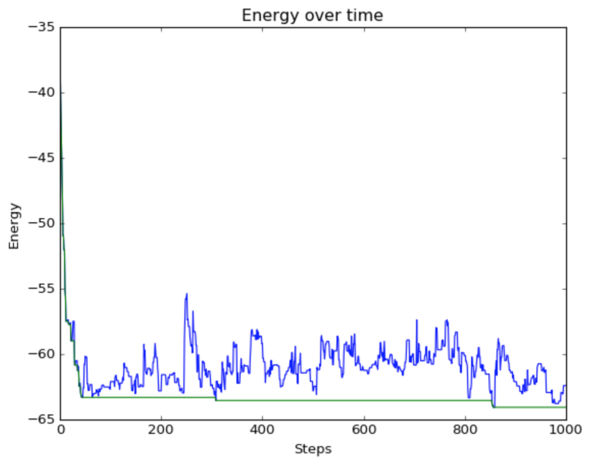

Figure 15: Best Energy v. Time

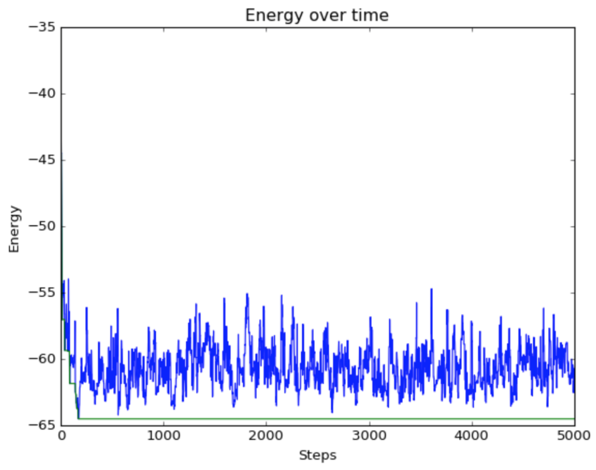

Figure 16: Long-range Stabilization T=1

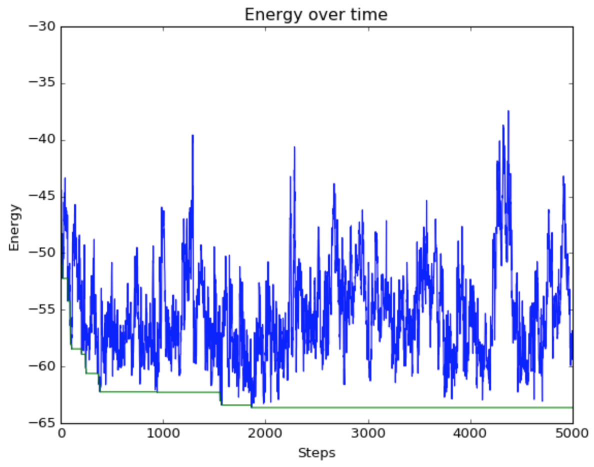

Figure 17: Long-range T=5

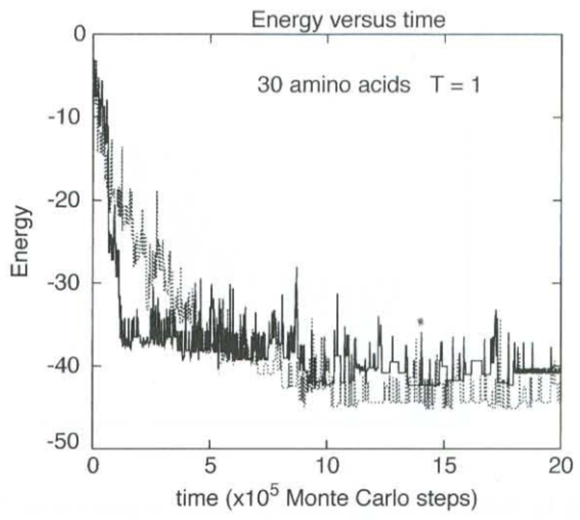

Figure 18: Stability for GN Model [1]

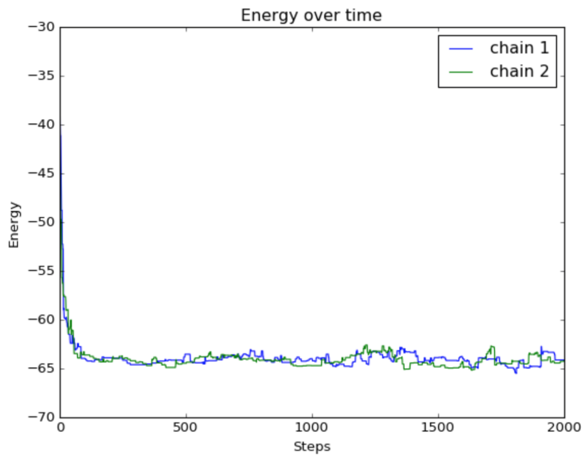

Figure 19: Stability for Repeated Runs at T=0.2

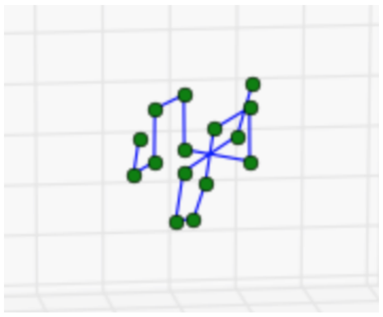

Figure 20: First Chain View 1

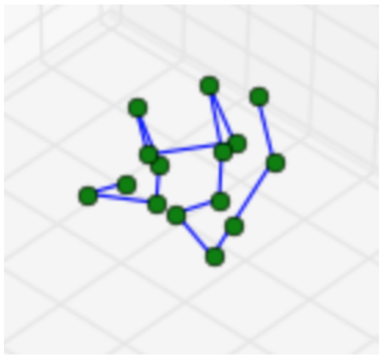

Figure 21: First Chain View 2

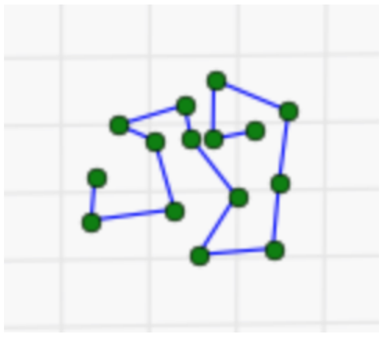

Figure 22: Second Chain View 1

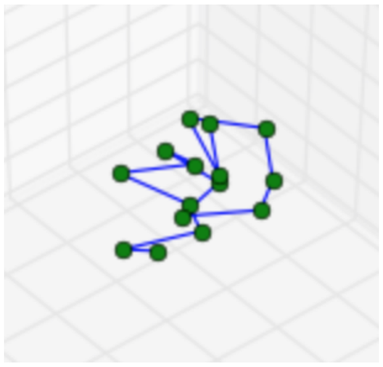

Figure 23: Second Chain View 2