CS 4414 Final Report
Team 5

Christopher Dare
cdare77@vt.edu
Virginia Tech
Computer Science, Mathematics

Dan Folescu
dan1345@vt.edu
Virginia Tech
Computer Science, Mathematics

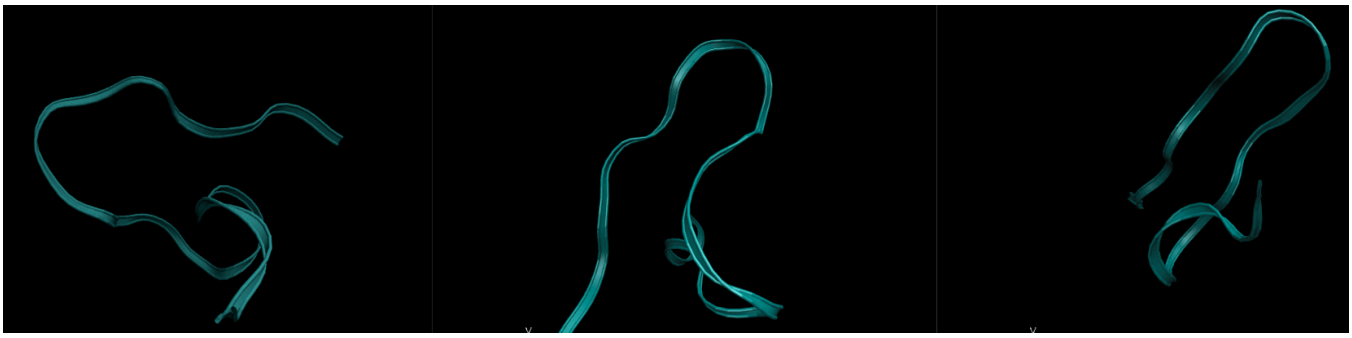


Figure 1: Folding configurations of Trp-cage under increasing Langevin collision frequency

ABSTRACT

In this report, our team examines computational methods for approximating the secondary structure of the mini-protein Trp-cage. Compared to most proteins, Trp-cage is a fairly small protein (only 20 molecules) and therefore allows much faster computation of state minimization techniques for our protein's overall potential energy. In particular, this paper presents a background on methods of numerical approximation and presents three iteration-based algorithms which attempt to optimize our minimization of the Trp-cage folding funnel.

1 INTRODUCTION

Protein folding is the process in which a protein (that is, a chain of amino-acids joined by peptide bonds) transitions to its intrinsic structure of lowest energy. The question of how a protein folds in its natural environment has been studied by biochemists for almost a century — however, research over the past several decades has shown that protein misfolding is a primary cause of certain diseases [5]. Consequently, the correlation between diseases and protein folding led to vast amounts of research on the subject.

The protein folding problem is by no means an easy or almost-solved problem. Originally studied in academia, the methodology of predicting folding patterns has changed dramatically due to the advent of scientific computing. [2]. Previously, researchers such as Cyrus Levinthal noted that the number of possible folding configurations was so large that it was impossible for a protein to test all possible configurations in the time experimentally observed — this became known as *Levinthal's paradox* [6]. Consequently, it was proposed that proteins must fold in a sequence of intermediate states, which are inherent to the protein itself; however, the problem of finding such intermediate states was nearly impossible to compute without the use of CPU's and GPU's.

Over time the advances in technology directly contributed to the ability to compute larger and larger protein folding patterns. However, due to the vast number of such patterns (as mentioned by Levinthal), the computational complexity of computing an exact solution for many proteins is still far too complex to solve for most modern computers. Therefore, methods and recent advances in the field of numerical approximation often find themselves applied to the field of biological computing, so that researchers are able to approach the correct structure of much larger proteins within a reasonable window of time.

Using numerical methods, researchers in the field are able to reduced the constraints on a problem and observe how a protein folds in a multitude of chemical environments. In fact, the manipulation of a protein's environment turns out to be an irreplaceable tool for finding the correct structure of a protein.

2 FOLDING LANDSCAPE

In general, almost no proteins arrange themselves into linear chains of atoms. The amino-acids which constitute a protein can arrange into a multitude of geometric shapes, and in particular one has that an amino-acid interacts with its surrounding neighbors and sometimes other distant amino-acids. Therefore, a protein always possesses some sort of potential energy corresponding to the interaction of its atoms with one another, either through chemical bonds or Van der Waals forces or other means. If only considering the bond-lengths, angles, torsion, electrostatic forces, and Van der

Waals forces, a typical potential energy function may look like:

$$\begin{aligned} U &= \sum_{\text{bonds}} k_b(b-b_0)^2 \\ &+ \sum_{\text{angles}} k_\theta(\theta-\theta_0)^2 \\ &+ \sum_{\text{torsion}} \sum_{n} k_{\phi,n}[1+\cos(n\phi+\phi_n)] \\ &= \sum_i \sum_{j>i} \frac{q_i g_j}{r_{ij}} \qquad \text{(Electrostatic)} \\ &= \sum_i \sum_{j>i} \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \qquad \text{(Van der Waals)} \end{aligned}$$

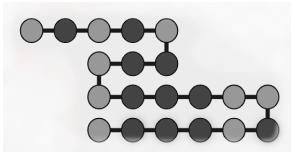
The problem may be simplified if one simply considering the lattice structure on hydrophobic-hydrophilic sequences, as pointed out by [4]. In general, interactions between hydrophobic and hydrophilic amino-acids almost always satisfy the following inequivalence relations:

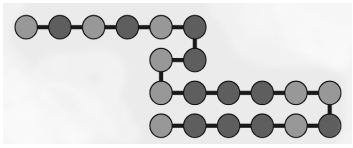
$$E_{PP} > E_{HP} > E_{HH}$$
$$2E_{HP} > E_{PP} + E_{HH}$$

where E_{PP} denotes the interaction energy between hydrophilic and hydrophilic, E_{HP} denotes the interaction energy between hydrophilic and hydrophobic, and E_{HH} denotes the interaction energy between hydrophobic and hydrophobic. Any choice of coefficients works, but we will stick to the choice of coefficients in [4] given by$E_{HH} = -2.3, E_{HP} = -1, E_{PP} = 0$. This leads to the following potential equation for a sequence of hydrophobic-hydrophilic amino-acids:

$$H = \sum_{i<j} E_{\sigma_i,\sigma_j} \Delta(\gamma_i - \gamma_j)$$

where γ_k indicates the position of an amino-acid in the lattice and $\Delta(\gamma_i - \gamma_j) = 1$ if and only if amino-acids at i and j are adjacent. If looking at the 2D lattice structure of our protein Trp-cage, we find local minima corresponding to the following lattice structures:





However, it is worth noting that our results later on will likely not look remotely close to the figures above since we are considering a protein in three dimensions. Moreover, the frequent bending patterns of our protein may indicate a torsion element which becomes present in higher dimensions (since there cannot exist any variation in the normal vector on a plane).

2.1 Folding Funnels

Though lattice structures are useful for predicting what a final secondary structure of a protein may look like, one often wants to observe all possible configurations holistically. Around the mid 90's, researchers such as Peter Wolynes devised a configuration space of all possible protein configurations in a similar fashion to how one considers the space of all particle trajectories in Hamiltonian mechanics. In the original paper [1], the authors describe how the configuration space of all protein configurations with respect to n variables is essentially the same as that protein's potential energy function with respect to the n variables. In other words, each protein has a unique energy potential function whose local minima represent intermediate folding states and whose global minimum represents the protein's intrinsic folding. Wolynes gives the following example in Figure 2

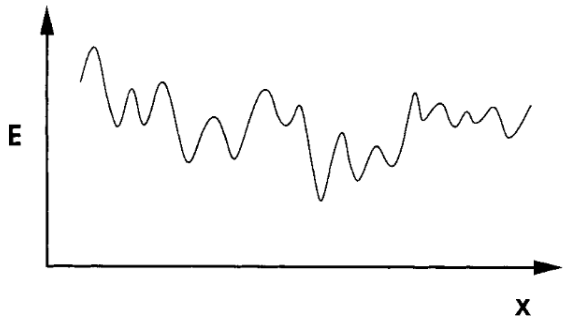


Figure 2: A 1-dimensional folding funnel of a protein varying over a single parameter (e.g. bond-angle)

Going back to our potential energy equation for U, we can see that there are far more parameters than a single independent variable, and thus we often observe higher-dimensional energy landscapes.

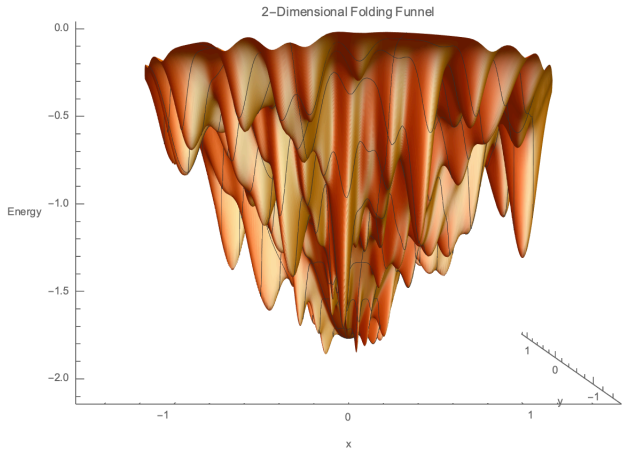


Figure 3: A 2-dimensional folding funnel embedded in $\mathbb{R}^3$

Though one may expect that this does not make a difference in terms of methodology, it turns out that certain numerical methods

tend to only work well with a high number (i.e. in lower dimensions) — therefore, we restrict our attention going forward to approximation methods which minimize manifolds (or point-data) of arbitrary dimension.

3 NUMERICAL APPROXIMATION

As previously mentioned, the problem of folding a protein can be extremely computationally expensive and is often impractical for larger protein structures. As we saw in Section 2, this problem can be reduced to finding the global minimum on the folding funnel — hence approximating the secondary structure is in fact equivalent to approximating the global minimum on the protein's respective folding funnel.

In this section, we explore three numerical methods which are commonly used for minimization problems:

(i) Random Search
(ii) Simulated Annealing
(iii) Nelder-Mead

We apply each numerical method to a smooth 2-dimensional function (so that the analytic solution may be easily computed) and hence compare cost and accuracy.

3.1 Random Search

As the name suggests, random search is a numerical minimization method which simply relies on randomness to eventually stumble upon the correct solution. An immediate benefit that one can observe of random search is that no two runs should be identical, and thus increasing the number of runs should increase the likelihood of finding the global minima. To clarify, the rationale behind the random search technique is much like the infinite monkey theorem, which states that if a monkey hits random keys on a typewriter for an infinite amount of time, it will eventually produce one of Shakespeare's plays. Again, we don't have an infinite amount of time, so one generally wants that each run finds the best local minima relative to the random start position.

In general, the algorithm for random search on a function $f : X \to Y$ can be broken down to the following:

Step A: Choose a random starting point $x_0 \in f(X)$ and fix some radius $r > 0$.

Step B: Assuming X is of dimension n and $rS^n_{x_0}$ denotes the re-scaling of S^n by a factor of r centered at x_0, put a uniform distribution on rS^n and pick a random point $y \in X \cap rS^n_{x_0}$. If $f(x_0) > f(y)$ move to the new position y — otherwise, sample a new point.

Thus, we begin sampling points around our initial random point until we find a neighboring position of lesser value, in which case we update our current position. As one should be able to tell, this method is prone to getting trapped in local minima if our radius $r > 0$ is not large enough. That is, if we get stuck in a minima such that the next minima is of distance $0 < r < R$ away, we will never move outside the current minima.

When testing random search on an arbitrary 2-dimensional folding funnel, we found that the accuracy was astonishingly low at 17%. Moreover, it took about 143 seconds to run 100 iterations on Mathematica 11.3, giving an average runtime of 1.43 seconds.

3.2 Simulated Annealing

Similar to the random search technique, simulated annealing relies on a distribution in order to predict which neighboring point to look at. However, simulated annealing is actually quite more complex than simply picking random points and evaluating the function locally. The minimization method originates from the practice of annealing metals — that is, heating and cooling metal in a fashion to maximize the strength of crystal bonds [3].

To motivate its algorithm, when annealing metals the smith will often try to cool the metal as slowly as possible after its been heated. While the metal is hot, the bonds are sporadic and may fluctuate between weaker or stronger formations — however, by cooling slowly the smith may increase the probability that the lattices align in a way that does not admit any cracks or weak points.

In a similar fashion, simulated annealing has a distribution $\lambda_t(f)$ representing the heat of our metal that slowly decreases as our time t increases. Much like random search, if we are at some position x_0 we look at a random neighboring point y; however, we no longer base our decision to jump if $f(x_0) > f(y)$ alone, but instead according to our distribution $\lambda_t(f)$ which factors in f at y. Since $\lambda_t(f)$ is a initially close to uniform, the movement of our current position is sporadic just like the crystals of a metal — this prevents us from getting stuck locally at some minima. However, as t grows we are much less likely to jump to a point y with $f(x_0) \leq f(y)$, hopefully bringing us closer to the globabl minimum.

Again, the algorithm for simulated annealing on a function $f : X \to Y$ with respect to some distribution λ_t can be broken down as follows:

Step A: Choose a random starting point $x_0 \in f(X)$ and fix some radius $r > 0$.

Step B: Take $t \to \infty$
  i) At each t, pick a random point $y \in X \cap rS^n_x 0$. Based on the distribution $\lambda_t(f)$, choose whether to stay at x_0 or move to y.

When testing simulated annealing on an arbitrary 2-dimensional folding funnel, we found that the accuracy was much better that random search at 63 %. Moreover, simulated annealing ran much faster on Mathematica 11.3, executing all 100 iterations in 43 seconds for an average runtime of 0.43 seconds.

3.3 Nelder-Mead

Our last minimization technique is slightly more technical than the previous two and thus the complete algorithms will not be fully covered since it does not offer any further insight into this paper's results.

From a general viewpoint, the Nelder-Mead technique essentially allows one to keep track of $(n + 1)$-points

$$x_1, x_2, \ldots, x_{n+1}$$

which are "somewhat" close to one another. As long as the points are distinct and no k points lie on the same $(k - 1)$-dimensional hyper-plane, one can construct a convex hull or simplex (i.e. the generalization of a triangle to n-dimensions). For example, the triangle is the 2-simplex and the tetrahedron is the 3-simplex constructed out of 4 triangles glued along their boundary. At each step, instead of moving toward the direction of our best point, we choose

to move away from our worst point — call it x_i. We then calculate the barycenter (i.e. center of mass) b_i of our remaining points and reflect x_i to a new point x_i' on the other side of b_i.

The reader should note that since these points are chosen to be "somewhat" close to one another, the Nedler-Mead method is fairly prone to getting trapped at local minima. When testing the Nelder-Mead method, we got the highest accuracy of 99% and second-best average runtime of 0.5 seconds.

4 METHODS

With all preliminary information covered, this section outlines the procedures our team used in attempt to find the correct folding of the protein Trp-cage:

ASN LEU TYR ILE GLN TRP LEU LYS ASP GLY GLY PRO SER SER GLY ARG PRO PRO PRO SER

Before outlining our team's methods used to approximate the secondary structure of Trp-cage, the reader should recall that the folding funnel of an arbitrary protein is often scattered with an enormous number of local minima which represent our protein's intermediate folding states. Consequently, one wishes to employ a numerical method which is not naturally prone to getting stuck in local minima. Of the three methods outlined above, this criteria indicates simulated annealing is the best method to approximate the minima of a folding funnel — therefore, we chose to utilize simulated annealing in all of our trials.

4.1 Protocol 1: Variation of Annealing Length

As described in Section 3, one expects that by cooling a material more slowly, the probability that a lattice structure properly aligns increases. Therefore, a natural followup question is whether the same principle applies when folding proteins.

For this experiment, all control variables are held constant except the temperature and the rate at which we cool our protein. In each trial, we heat our protein to its melting temperature of 317K in about 0.2 ns and proceed to heat it to a maximum of 400K in 1 ns. From there we cool our protein from 400K to 350K in X ns and then back to the melting temperature of 317K in Y ns. Our values of X and Y for each trial are as follows:

Trial	X	Y
1	2 ns	2 ns
2	3 ns	2 ns
3	2 ns	3 ns
4	3 ns	3 ns

Based off our hypothesis, we expect that the minimum potential energy will decrease from trial 1 to trial 4. By constructing a simple BASH script which runs four distinct input files through AMBER (corresponding to our four trials), we are able to monitor the potential energy as a function of time while we raise and lower the temperature.

4.2 Protocol 2: Effect of Repeated Annealing on Minimization

The section above on variation of annealing lengths indicates that it is best to run simulated annealing at longer lengths in order to achieve an optimal local minima. However, this raises the question of whether our minimization process benefits from multiple simulated annealing runs. If this is the case, our protein will need less and less energy to break out of local minima since it is gradually approaching the global minimum. Therefore, the temperature to which we heat the protein should gradually decrease in each consecutive annealing run. For this particular experiment, we define a single "multiple annealing" run as the following:

(1) Use SANDER to shift the protein to the nearest minima
(2) Heat the protein to its melting point (317K)
(3) Hold the protein in equilibrium at its melting point for 20 ns
(4) Run simulated annealing at a maximum temperature of 400K for 6 ns
(5) Run simulated annealing at a maximum temperature of 390K for 6 ns
(6) Run simulated annealing at a maximum temperature of 380K for 6 ns
(7) Run simulated annealing at a maximum temperature of 370K for 6 ns
(8) Run simulated annealing at a maximum temperature of 360K for 6 ns

Instead of resuming each simulated annealing at the last state of the previous annealing, we calculate the frame of minimum potential energy (sufficiently far after the heating phase) and use that as input — the hope is that this will ensure our protein is on the correct trajectory to the global minimum.

4.3 Protocol 3: Effects of Langevin Collision Constant on Minimization

Each AMBER run which we have undertaken has been done in the setting of Langevin dynamics — that is, a modeling approach which tries to replicate the viscosity of solvents present during protein folding in real life. In AMBER, we can alter such dynamics through two variables: temperature and γ_{ln} (the Langevin damping constant).

Since our team's previous trials were all dependent on temperature alone, we chose to focus our final protocol in terms of viscosity. Therefore, for a fixed value of γ_{ln} we define a trial for our third protocol to be a run at the following:

(1) Use SANDER to find local minima in EPTOT.
(2) Heat protein to 320K(7200 PS)
(3) Hold protein in equilibrium at 320K(7200 PS)
(4) Cool to 100K(7200 PS)

In particular, our team only runs three trials corresponding to γ_{ln} values $\gamma_{ln} = 0.001$ (which we define to be "low viscosity"), $\gamma_{ln} = 1$ (which we define to be a "medium viscosity"), and $\gamma_{ln} = 5$ (which we define to be a "high viscosity").

5 RESULTS

In this section we further describe each protocol executed, present graphical representations of our results, and attempt to describe the nature of such results.

5.1 Protocol 1: Variation of Annealing Length

As previously described in Section 4.1, we predict that by increasing the length of annealing our protein will reach a better minimum. That is, we expect a direct correlation between length of annealing and energy minimization. After running the four trials described in Section 4.1 above, our team found the following results:

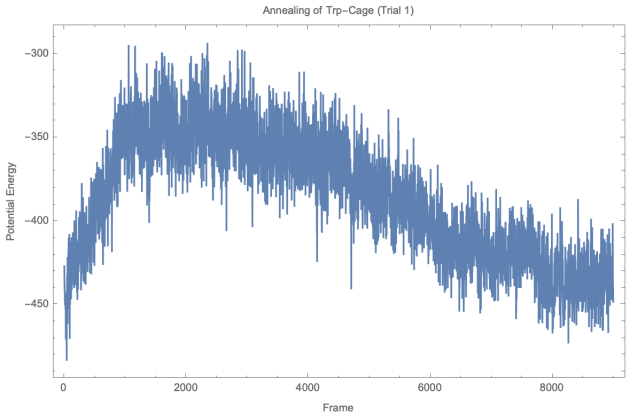


Figure 4: (Protocol 1, Trial 1): Minimum Energy -483.339

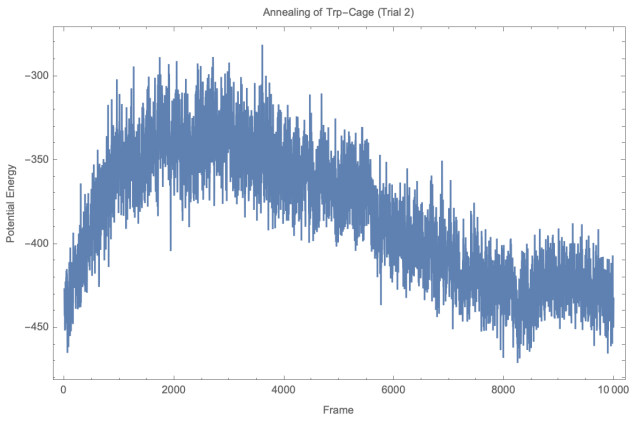


Figure 5: (Protocol1, Trial 2): Minimum Energy -470.837

Before interpreting the results, one should note that the minimum of trial 1 occurs at the very beginning stages before even heating the molecule. Since the same SANDER + heating output is provided as input to each trial and each trial is heated to 400K at the same rate, this minima is likely caused by a degree of randomness. By chopping off the first 75 frames of trial 1, we see that a more accurate minima occurs at frame 8264 of energy -472.978. Therefore,

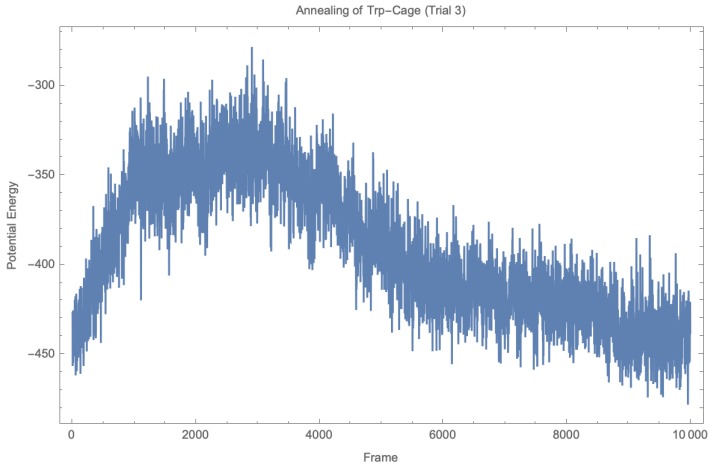


Figure 6: (Protocol 1, Trial 3): Minimum Energy -477.864

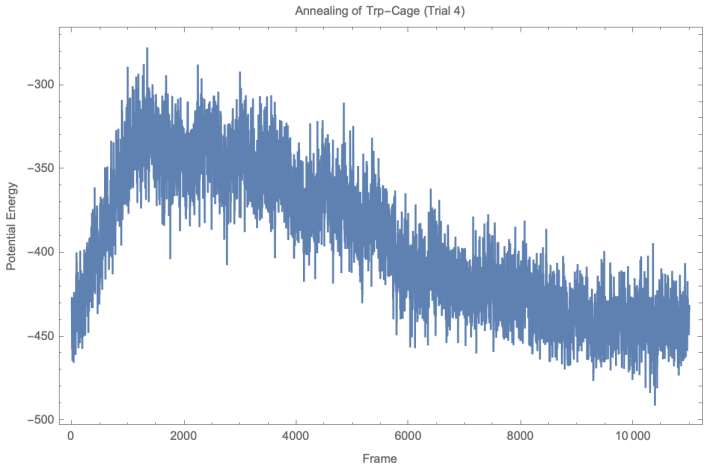


Figure 7: (Protocol1, Trial 4): Minimum Energy -490.976

it is easy to see that there is a correlation between increasing the annealing time of our protein and the minimum energy state after annealing.

Lastly, we visualize how our protein folds across all four annealing trials using VMD's ribbon representation in Figures 8 through 11 below.

Clearly, the slight alterations in a simulated annealing run can lead to vastly different results in terms of the structure of a protein. We are able to see, however, that there is a significant torsion element present in the first seven amino-acids (asparagine through leucine) for each trial, indicating that global minimum (i.e. intrinsic secondary structure) has a torsion element along this chain as well.

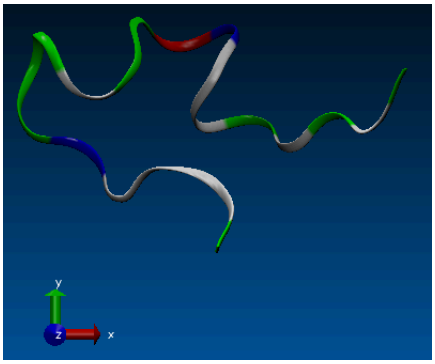


Figure 8: Minimal Configuration of Trial 1 (Protocol 1)

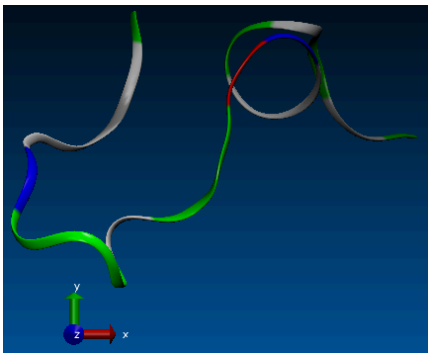


Figure 9: Minimal Configuration of Trial 2 (Protocol 1)

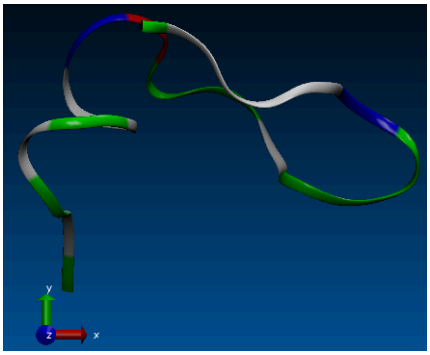


Figure 10: Minimal Configuration of Trial 3 (Protocol 1)

5.2 Protocol 2: Effect of Repeated Annealing on Minimization

Since the overall objective of these experiments is to find the secondary structure of Trp-cage, we choose to use the optimum annealing length from Protocol 1 to ensure the best results. That is, we define a single annealing 'run' as heating the protein to some temperature $X > 317K$, cooling to a new value Y in 3 ns, and then cooling back down to $317K$ in 3 ns.

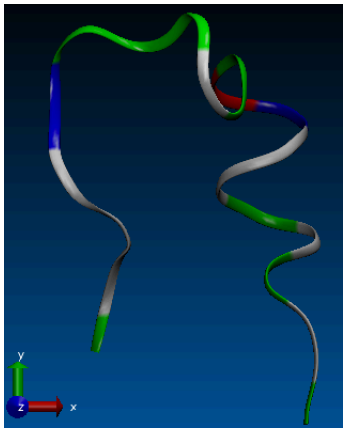


Figure 11: Minimal Configuration of Trial 4 (Protocol 1)

Assuming we use some text processing script (which in our case is the `process_mdout.perl` script adapted from AMBER's tutorial) to enumerate our frames by potential energy, we may convert our previous minimal energy frame to input using the following:

```
echo '$AMBERHOME/bin/cpptraj
../minip.prmtop << EOF' >> ptraj_1.sh

cat summary.EPTOT | gawk
'BEGIN{min=1000;minidx=-1} {if
($1>1000 && $2<min) {min=$2; minidx=$1}}
END {print minidx}' | grep -m 1 -f -
../annealing1.out | gawk
'{a = $3 / 1000; print "trajin
../annealing1.mdcrd", a, a}' >> ptraj_1.sh

echo 'trajout min_run1.rst restart' >> ptraj_1.sh
chmod +x ptraj_1.sh
```

The reader should take note of the following condition in our code above:

```
$1>1000 && $2<min
```

The first number (in our example `summary.EPTOT`) is precisely the frame number, while the second number is our energy potential value. The condition that the only frames we consider are after the first 1000 ensures that the minima we record occurs *after* our annealing, and not as some random perturbation at the beginning; this will allow us to accurately measure how effective multiple runs of annealing truly is.

Our team ran exactly four of the "multiple annealing" trials, as outlined in Section 4.2. For the sake of clarity, we will not show the 20 energy potential plots associated to each single annealing run; instead, we will present how the minimum potential energy state of each annealing run varied within each trial. For the first trial, we observed the minimum energy state fluctuate as depicted in Figure 12 as we increased the number of times annealing was applied.

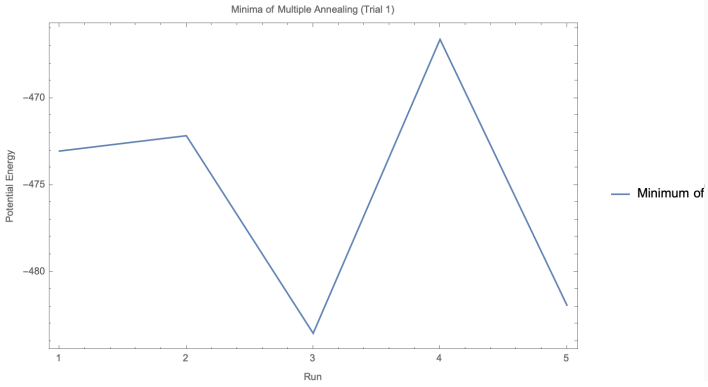


Figure 12: (Protocol 2, Trial 1): Minimum Energy Achieved vs. Annealing Run

One can easily recognize that there is no strong correlation between consecutive annealing and minimization based on the data of the first run. In fact, we get very similar results for runs 2-4:

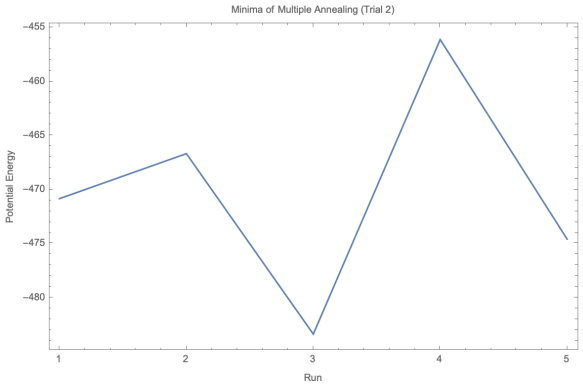


Figure 13: (Protocol 2, Trial 2)

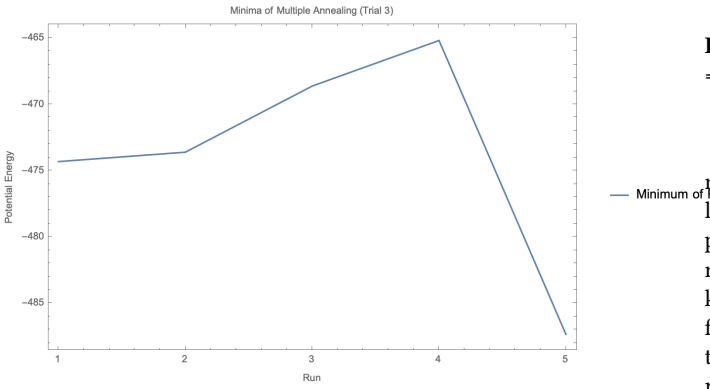


Figure 14: (Protocol 2, Trial 3)

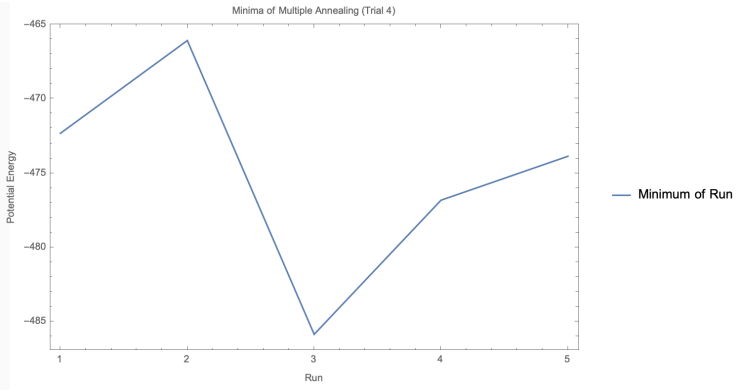


Figure 15: (Protocol 2, Trial 4)

Therefore, we conclude that there is no evidence which shows consecutive annealing achieves a better minimum that a single simulated annealing run.

5.3 Protocol 3: Effects of Langevin Collision Constant on Minimization

We have the following results for tests on low (Figure 16), medium (Figure 17), and high (Figure 18) relative gamma constants.

We also include the plots of energy potentials for low, medium, and high γ_{ln} values in figures 19, 20, and 21, respectively.

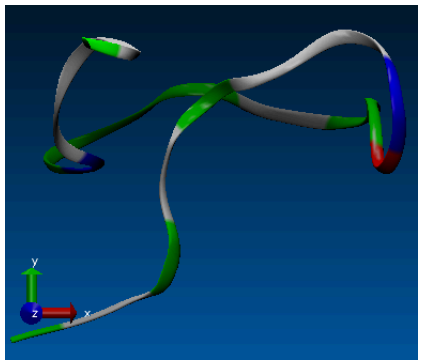


Figure 16: Low Gamma($\gamma_{ln} = 0.001$) Configuration - EPTOT = -600 kcal/mol

5.3.1 Optimized Langevin Constant - Longer Run. From experimentation, we see that an optimal gamma constant is not too low, lest there not be enough interaction with the protein to adequately preform molecular dynamics on our time scale. Through experimentation by Professor Onufriev's lab(No current reference), we know that the optimal gamma constant is also not too high and, from a computational/dynamic perspective, is actually 0.01. We use this gamma constant, as well as heating/equilibrium/cooling experience from the previous tests to create the following protocol:

(1) Use SANDER to find local minima in EPTOT.
(2) Heat protein to 320K(25000 PS)
(3) Hold protein in equilibrium at 320K(25000 PS)

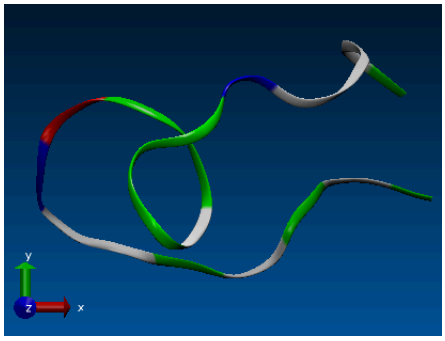


Figure 17: Mid Gamma($\gamma_{ln} = 1$) Configuration - EPTOT = -620 kcal/mol

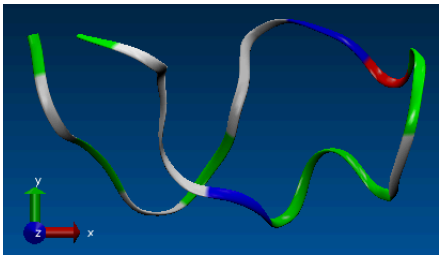


Figure 18: High Gamma($\gamma_{ln} = 5$) Configuration - EPTOT = -620 kcal/mol

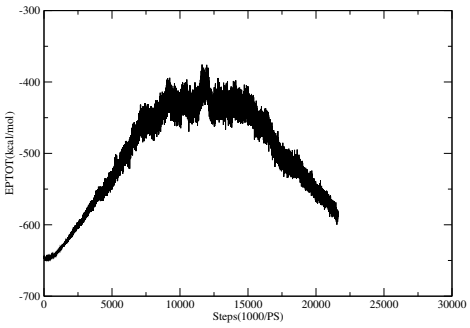


Figure 19: Low Gamma EPTOT vs Steps

(4) Cool to 0K(50000 PS)

From this protocol, we have energy potential seen in Figure 22, and the protein structure seen in 23.

6 CONCLUSION

In summary, our team tested three different methods to minimize the potential energy of Trp-cage's folding funnel and thus approximate the protein's secondary structure:

(1) Increasing the cooling time of simulated annealing

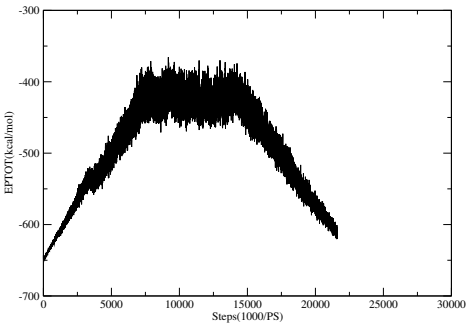


Figure 20: Mid Gamma EPTOT vs Steps

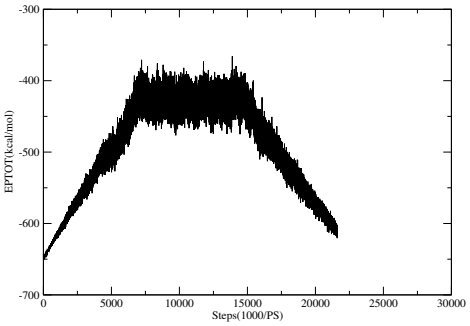


Figure 21: High Gamma EPTOT vs Steps

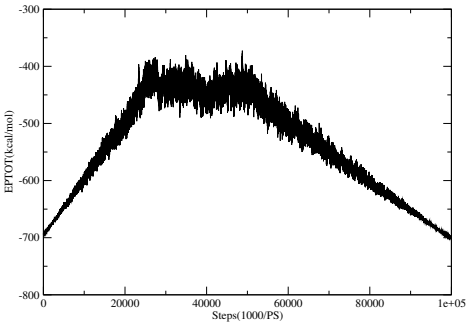


Figure 22: Optimal Gamma EPTOT vs Steps

(2) Run consecutive annealing tests at decreasing maximum temperatures

(3) Decrease the viscosity of the environment

From the results of our first experiment, we found that there's a direct correlation between how long a protein cools in simulated

annealing and the minimum energy state the protein settles in. In particular, this affirmed our hypothesis that simulated annealing on protein structures behaves similarly to annealing metals.

In our second experiment, we ran several "multiple annealing" trials to test whether there was any correlation between consecutive annealing and minimization of potential energy. However, by mapping the minimum energy of each annealing run, we were able to find that there is indeed no strong correlation between the two. In particular, this implies that minimization does not necessarily benefit from running an algorithm multiple times, but instead by running longer, more accurate trials.

Lastly, we attempted to run our simulated annealing at several viscosity values to see whether this had any affect on the protein's potential energy. As one may expect, particle collisions increase a protein's total energy and therefore cause more sporadic protein behavior. Therefore, we found that lower viscosity values led to much better minimization results.

Ultimately, our lowest potential energy was achieved via simulated annealing in a solution of viscosity $\gamma_{ln} = 0.01$, yielding exactly -701 kcal/mol and the following secondary structure in Figure 23

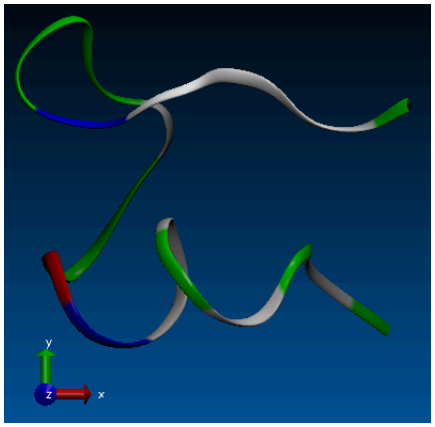


Figure 23: Best Secondary Structure Approximation

7 PROJECT ASSESSMENT

In conclusion, this project presented an opportunity to engage with material that most (if not all) of the class had not seen previously. Our team was able to run numerous experiments on VMD software, as well as new equipment (e.g. Virginia Tech ARC supercomputer).

Due to the diverse range of students' experience with computational tools (e.g. BASH scripting, awk, vim), some of the preliminary material seemed repetitive. If this course were restricted to computer science majors alone, we feel that we would have been able to get into much more relevant material.

It was interesting to learn about various numerical methods, both in their boons and shortcomings. Contrary to our paragraph above, this was likely a topic that was repetitive for some applied math majors yet essential for many other students.

Overall, our team found this capstone project to be an enjoyable experience.

REFERENCES

[1] Joseph D. Bryngelson, José Nelson Onuchic, Nicholas D. Socci, and Peter G. Wolynes. 1995. Funnels, Pathways, and the Energy Landscape of Protein Folding: A Synthesis. *PROTEINS: Structure, Function, and Genetics* 21 (1995), 167–195. https://doi.org/10.1002/prot.340210302

[2] Ding Chen Y, Nie H F, and et al. 2007. Protein Folding: Then and Now. *Arch Biochem Biophys* 469, 1 (2007), 4–19.

[3] A. Das and B. K. Chakrabarti. 2005. Quantum Annealing and Related Optimization Methods. *Lecture Note in Physics* 679 (2005).

[4] Hao Li, Robert Helling, Chao Tang, and Ned Wingreen. 1996. Emergence of Preferred Structures in a Simple Model of Protein Folding. *Science, New Series* 273, 5275 (1996), 666–669.

[5] Enrique Reynaud. 2010. Protein Misfolding and Degenerative Diseases. *Nature Education* 3, 9 (2010), 22.

[6] R. Zwanzig, A. Szabo, and B. Bagchi. 1992. Levinthal's Paradox. *Proc Natl Acad Sci USA* 89, 1 (1992), 20–22.

8 CODES DEVELOPED

```
   langevin dynamics simulations
&cntrl
       ntx = 1, irest=0,
       imin = 0, nstlim = 100000000,
       dt = 0.001, ntt = 3, gamma_ln=0.01,
       temp0 = 100.0, tempi=0.0,
       ntc = 2, ntf = 2,
       ig=-1,
       igb=8, ntb = 0, saltcon=0.145,
       ntwx = 1000, ntwe = 0,
       ntwr = 1000, ntpr = 1000,
       cut = 999.0, rgbmax = 999.
       nmropt = 1,/
&wt type='TEMP0',
istep1=0,istep2=25000000,
  value1=0.0, value2=320.0/
&wt type='TEMP0',
istep1=25000001,istep2=50000000,
  value1=320.0, value2=320.0/
&wt type='TEMP0',
istep1=50000001,istep2=100000000,
  value1=320.0, value2=0.0/
&wt type='END'/
```

Figure 24: Optimal Gamma Test Code

```
   langevin dynamics simulations
&cntrl
       ntx = 1, irest=0,
       imin = 0, nstlim = 43200000,
       dt = 0.0005, ntt = 3, gamma_ln=0.001,
       temp0 = 100.0, tempi=0.0,
       ntc = 2, ntf = 2,
       ig=-1,
       igb=8, ntb = 0, saltcon=0.145,
       ntwx = 1000, ntwe = 0,
       ntwr = 1000, ntpr = 1000,
       cut = 999.0, rgbmax = 999.
       nmropt = 1,/
&wt type='TEMP0',
istep1=0,istep2=14400000,
  value1=0.0, value2=320.0/
&wt type='TEMP0',
istep1=14400001,istep2=28800000,
  value1=320.0, value2=320.0/
&wt type='TEMP0',
istep1=28800001,istep2=43200000,
  value1=320.0, value2=100.0/
&wt type='END'/
```

Figure 25: Gamma Test Code Prototype