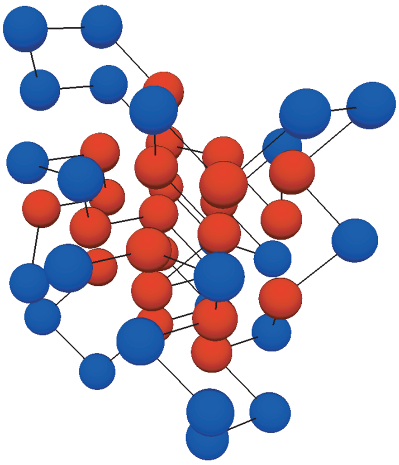
CHALMERS UNIVERSITY OF GOTHENBURG



Protein Folding

Implementation of the Simulated Annealing Algorithm on Simple Three-Dimensional Models

Bachelor's Thesis

LINUS B ÖRJESSON
OSCAR KALLDAL
MAXIMILIAN LUDVIGSSON
JOHNNY NGU
ANDREAS NILSSON
GUSTAV ÖHMAN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2014

Protein Folding
Implementation of the Simulated Annealing Algorithm on Simple Three-Dimensional Models

LINUS BÖRJESSON
OSCAR KALLDAL
MAXIMILIAN LUDVIGSSON
JOHNNY NGU
ANDREAS NILSSON
GUSTAV ÖHMAN

Examiner: Prof. Jan Skansholm
Supervisor: Prof. Graham Kemp

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Visual representation of a folded sequence obtained using our simulated annealing algorithm on a face-centered cubic lattice.

Department of Computer Science and Engineering
Gothenburg, Sweden June 2014

Abstract

How an arbitrary coil of amino acids folds into its functional structure is known as *the protein folding problem*. Since the underlying mechanisms that guide protein folding in nature are widely unknown, simplified models are studied. Many of these models have energy levels as the focal point in order to find the native state and may ignore other relevant constraints. While these simplified models may seem too trivial to have any resemblance to the physical reality, they can be used to explore concepts and ideas that may lead to further insights on how proteins fold.

This thesis studies the use of simulated annealing optimization techniques to find low energy states in simple lattice and off-lattice models. A certain emphasis is placed upon looking for patterns in the results emerging. One simple off-lattice model and two lattice models are considered, a cubic lattice and a face-centered cubic lattice. Compared to the optimal energy, low energy conformations of 48-residue chains are found in reasonable time. It is concluded that while the method can not be said to exhibit the behavior of finding one consistent native state each time it is run, patterns do emerge in the results.

Sammanfattning

Hur en godtycklig kedja av aminosyror veckas till sin funktionella struktur kallas *proteinveckningproblemet*. Eftersom de bakomliggande mekanismerna som styr proteinveckning i naturen är okända studeras förenklade modeller. Många av dessa modeller fokuserar på energinivåer för att hitta proteinets naturliga tillstånd och ignorerar många andra aspekter. Även om dessa förenklade modeller kan tyckas alltför triviala för att ha likheter med den fysiska verkligheten kan de användas för att utforska begrepp och idéer som kan leda till ytterligare insikter om hur proteiner veckar sig.

Denna avhandling studerar användningen av simulerad-glödgningsoptimeringstekniker för att hitta låga energitillstånd i enkla modeller. Den studerar om dessa tekniker kan sägas likna en simulering av hur ett protein veckas, med viss tonvikt att leta efter mönster i resultaten uppkomna som en bieffekt av optimeringsmetoden som används. En enkel kontinuelig modell och två gittermodeller används, ett kubiskt gitter och ett ytcentrerat kubiskt (förkortat FCC på engelska) gitter. För kedjor av 48 aminosyror fanns låga energikonformationer jämfört med optimum inom rimlig tid. Slutsatsen drogs att även om metoden inte kan sägas uppvisa beteendet att hitta ett och samma naturliga tillstånd varje gång den körs, dyker vissa mönster upp i resultaten.

Acknowledgements

Graham Kemp, for the support and for providing the idea for the novel off-lattice model.

Maryana Wånggren, for pulling Graham down to earth.

Group 35, Gothenburg 2014-05-19

Contents

1

Introduction

The manner in which proteins, chains of amino acid residues, acquire their three-dimensional conformation has long been a subject of inquiry and many different explanations of the underlying mechanism have been proposed. One such hypothesis, the so-called *thermodynamic hypothesis*, states that given a certain set of conditions, such as pH, ionic strength and temperature, a protein's native conformation is that at which the Gibbs free energy is the lowest [1]. Cyrus Levinthal, a prominent American molecular biologist, noted that this leads to an interesting problem, namely *Levinthal's paradox*. Quite simply, how can the proverbial needle that is the lowest energy conformation be found in the haystack of possible conformations in such a consistently short timespan [2]?

Solving Levinthal's paradox is achieved by the concept of folding funnels. A protein's *energy landscape* has a more or less direct slope toward its native conformation, often likened to a ski slope. The protein's folding pathway, the path it takes to get to its native state, varies in a manner analogous to a skier's different descents down the same slope. Thus, simply finding the lowest energy conformation is not the only aspect of interest when considering protein folding [2].

When simulating protein folding in silico, certain simplifications are made, which entail disregarding certain facets of the biological and chemical reality. The driving forces behind the folding of proteins is a widely researched topic, and exactly how the process works is still a subject of debate, calling for the need to explore a variety of simplified models [3]. One family of simplified models consider the protein as a sequence of residues placed on a lattice, constraining the position of each amino acid residue to a discrete set of points. This allows for further simplifications concerning the energy of a protein's conformation as well as the kinetics guiding its folding. Much research has been put into exploring these lattice models in order to gain insight into the mechanisms responsible for protein folding in nature [4].

The advent of computer technology has made in-depth study of different aspects and models of protein folding possible through advanced simulation techniques [5]. Particular emphasis has been placed upon Monte Carlo methods when designing algorithms to identify the lowest energy conformation [6]. One such algorithm commonly applied is the Metropolis-Hastings algorithm, frequently used when simulating complex, nonstandard multivariate distributions [7].

Though it appears at first to be trivial in nature, solving the problem of protein folding has a myriad of real world applications. When proteins misfold and are not corrected or destroyed by the cell's internal regulatory systems, they have a tendency to form fibrous aggregates known as amyloids. These abnormal structures sometimes accumulate in sensitive structures, such as the brain, where they cause neurodegenerative diseases including Alzheimer's and Huntington's. Protein misfolding can also potentially form prions, which are proteins of abnormal conformations that possess the ability to misfold other proteins of the same type. This cascade causes prion diseases such as Creutzfeldt-Jakob disease, bovine spongiform encephalopathy (mad cow disease), and scrapie in sheep [8]. A deeper understanding of the underlying mechanism could alleviate the suffering of millions, both animals and humans.

There are additional implications in the world of industrial biotechnology. The ability to successfully predict the conformation of a protein would aid significantly in protein engineering, both in manipulating existing proteins, to increase thermostability, for example, and in creating completely novel proteins via de novo synthesis [9].

1.1 Aims

This thesis sought to study protein folding by using a simulated annealing method and evaluate how efficiently it finds low energy conformations in simple protein models. The simulated annealing method studied is commonly used to solve optimization problems. Rather than just finding a low energy conformation, it performs a local search and produces a sequence of geometrically similar conformations acting as a pathway from the unfolded to the folded state. This method was further investigated to ascertain whether it affects which low energy states are found and if it is possible to distinguish if certain states are preferred for reasons other than their low energy.

1.2 Methodology

The task of recreating protein folding through the use of a computerized algorithm was broken down into four tasks: initial creation and testing of the algorithm using a cubic, three-dimensional lattice; subsequent testing of the algorithm on a face-centered cubic lattice; modifying the algorithm in order to simulate protein folding off-lattice; and creating a means of displaying the resulting fold using a graphical user interface. These areas are expounded upon in section 3.

The optimization framework built up by the simulated annealing algorithm was then evaluated on the protein models. Its performance was assessed by comparing the low energy states to known global minima obtained by HPstruct, a simulation tool [10]. Similarity of the resulting conformations were investigated with statistical methods to evaluate whether patterns could be seen in the data.

1.3 Scope

Many biochemical processes were disregarded due to the added complexity of simulating them. However, as the program was constructed to be easily amendable and extensible, future studies could incorporate these interactions in the model. One simplification was to consider the protein in the simulation process as a lone entity in space, ignoring the possibility of other affecting nearby structures. Additionally, in the two lattice models, the amino acids were limited to discrete points in space.

The spatial conformations adopted by the protein while undergoing folding was limited by only being able to change through a set of predefined moves. Furthermore, the complexity that arose from the properties of the amino acids was reduced by categorizing the amino acids as either hydrophobic or polar in accordance with the *hydrophobic-polar model*. In nature, the length of a protein varies substantially. In this thesis, only amino acid chains of length 48 were considered.

1.4 Report overview

The report is divided such that the underlying theory needed to understand the protein folding problem, and to understand the implementation of the tools and algorithms used, is presented first. The theory concerning the underlying biology, presented in chapter 2, may be read cursorily, but its contents are important in order to appreciate why the simplifications are made. In chapter 3, the software implementations used are described in greater detail, with an emphasis on the original work done. In the last three sections, the results of simulations using the implemented software are presented, analyzed, and discussed.

2

Theory

THE PROBLEM OF PROTEIN FOLDING is interdisciplinary as it necessitates understanding of the fields of chemistry, biology, mathematics, and computer science [11]. A protein's structure is determined by the interactions of its constituent amino acids. Understanding of both the underlying chemistry and the optimization algorithms used is therefore of critical importance in order to transfer the biological problem of protein folding to a computational one.

2.1 Biology of proteins

The central dogma of molecular biology, first postulated by Crick in 1958, explains the manner in which genetic information flows in all organisms, including everything from bacteria to humans. The genomic sequence, consisting of DNA, is first transcribed to RNA whereupon it is translated to proteins [12]. The codons in RNA, consisting of nucleotide triplets, relay instructions to the ribosomes that aid in translation by signaling which amino acid to incorporate into the growing polypeptide chain. As the constitution of these triplets varies, so does the sequence of amino acids and the resulting protein produced [8, 9].

2.1.1 Amino acids

Amino acids are organic compounds that contain a carboxyl as well as an amine group. There are 22 proteinogenic amino acids [13], of which 20 are commonly found in humans [8, 14]. A condensation reaction, shown in figure 2.1, is responsible for creating the linkage between the amino acids [15]. The resulting covalent peptide bond is the reason that proteins are oftentimes referred to as polypeptides [8].

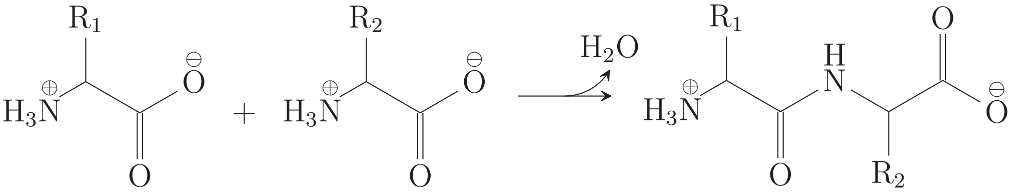


Figure 2.1: Condensation reaction between two generic amino acids [15]. R_1 and R_2 indicate side chains which vary between the different amino acids. These are not involved in the reaction and free to interact after the polypeptide linkage is formed.

The different properties of amino acids are due to the variation of the side chains, denoted by R_1 and R_2 in figure 2.1. These do not take part in creating the polypeptide backbone of the protein and are generally divided into four different categories based upon their charge at neutral pH: acidic (negative), basic (positive), uncharged polar, and nonpolar [8], as seen in table 2.1. Nonpolar amino acids are considered hydrophobic as they are less soluble in water than polar residues, which can form hydrogen bonds with water molecules [14].

Table 2.1: Proteinogenic amino acids commonly found in humans [8].

Polar			Nonpolar	
Acidic	Basic	Uncharged		
aspartic acid	lysine	asparagine	alanine	valine
glutamic acid	arginine	glutamine	leucine	isoleucine
	histidine	serine	proline	phenylalanine
		threonine	methionine	tryptophan
		tyrosine	glycine	cysteine

2.1.2 Protein structure

The conformation of a protein is described at four different levels: primary, secondary, tertiary, and quaternary structure [13]. The primary structure is the sequence of amino acid residues that makes up the polypeptide. Said sequence is given in order from the amine group of the N-terminus to the carboxyl group of the C-terminus, shown on the left and right side of figure 2.1 respectively. This convention is due to the manner in which amino acids are synthesized in nature.

The secondary structure is a means of describing local conformational units that are stabilized in large part through hydrogen bonding. The commonly occurring structural motifs are: α-helices, β-strands/β-sheets, turns, and coils. Of these, α-helices and β-sheets are the most ubiquitous. These can clearly be seen in the ribbon diagram in figure 2.2 where the red spirals represent α-helices and the blue arrows show β-sheets.

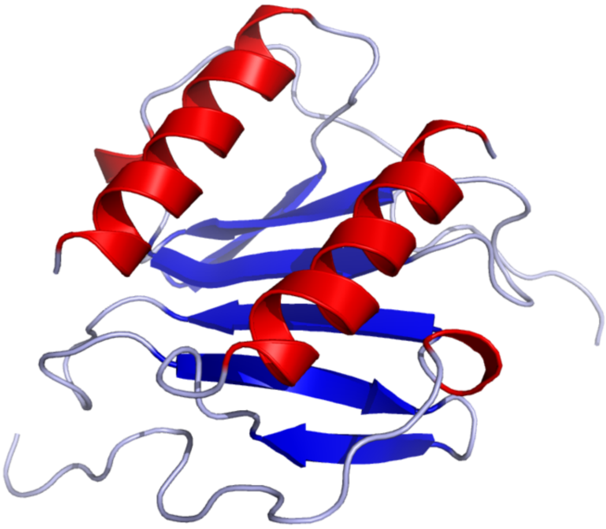


Figure 2.2: Ribbon diagram showing the solution structure of human interleukin 8. The red spirals indicate α-helices while the blue arrows show β-sheets. Both of these structures are components of a protein's secondary structure.

A protein's tertiary structure refers to the polypeptide's full three-dimensional conformation. This structure is comprised of secondary structural units held together by long-range electrostatic and hydrophobic interaction as well as hydrogen bonds and van der Waals forces. As proteins can be built up of more than one polypeptide subunit, quaternary structure is used to describe how these interactions occur and the resulting conformation of the protein.

2.2 Folding theories

According to the *thermodynamic hypothesis* first posited by Anfinsen in 1973, a protein adopts the conformation at which the Gibbs free energy of the system is lowest, the so-called native state. Given a specific set of environmental parameters, such as solvent composition, pH, and temperature, a protein should in theory adopt the same tertiary structure regardless of its initial conformation. The observation that denatured proteins return to their native state lends credence to Anfinsen's theory, as do simulations run with computer models [16].

There are three different theories that seek to explain the driving force behind protein folding. These theories are referred to as the framework model, the hydrophobic collapse model, and the nucleation condensation model [13]. The differences between these theories lie primarily in the significance they place upon long-range and local interactions.

According to the framework model, local interactions cause the protein to adopt secondary structure before long-range interactions form the tertiary structure. This theory relies upon the idea that intramolecular hydrogen bonding plays the greatest role in causing proteins to fold. This notion is refuted by the fact that a hydrogen bond between a solvent water molecule and an amino acid is more energetically favorable than a hydrogen bond between two amino acids. However, hydrogen bonding within the core of the protein, where the amino acids are shielded from water molecules, may potentially stabilize proteins [3].

The hydrophobic collapse model supposes that a protein adopts its tertiary conformation immediately after synthesis due to the strength of its hydrophobic interactions. Upon collapse, the protein adopts a *molten globule* conformation allowing for long-range interaction to form bonds, after which local interactions form secondary structure in the form of α-helices and β-sheets. This hypothesis arose from the observation that nonpolar amino acids in solvent tend to congregate in a manner akin to an oil droplet dispersed in water [17]. Whether this is due to an attractive force between the nonpolar amino acids or a repulsive force imparted by the surrounding water remains a topic of debate [3].

The third theory, the nucleation condensation model, can be viewed as a compromise between the aforementioned proposals. It supposes that collapse of the protein chain allows for local and long-range interaction simultaneously, causing the protein to fold without adopting any intermediate conformations.

2.3 Optimization strategy

Many optimization problems are too complex to solve exactly, as the time required to compute a solution in certain cases increases exponentially with the problem's size. These problems are usually solved with algorithms aiming to find near optimal solutions instead of finding the actual optimum. By viewing the protein folding problem as a simple optimization problem seeking to minimize the conformational energy, it is possible to adapt the problem model to the framework of existing optimization algorithms.

2.3.1 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a type of Markov chain Monte Carlo (MCMC) algorithm, that is to say an algorithm that uses Markov chains to simulate random sampling from a target probability distribution $\pi(\cdot)$ [18]. A Markov chain is a discrete stochastic process where, given a state at a specific time, all future states are independent of the previous ones and depend solely upon the current state. It can thus be described as *memoryless* since the state at a given time only depends only upon the immediately preceding one.

The basis of the algorithm is the notion that if a Markov chain X_t, $t = 1, 2, 3 \ldots$, having $\pi(\cdot)$ as its stationary distribution is found, it can be run for a sufficient number of time steps after which all states of the chain will approximate random states drawn from the target distribution $\pi(\cdot)$. To simulate such a Markov chain, the following scheme is used. At each state X_t at time t, a candidate state Y is drawn from a candidate distribution $q(\cdot|X_t)$, which can easily be sampled. This candidate is then tested against a so-called *Metropolis criterion* where it has a probability of

$$\alpha(X,Y) = min\left(1, \frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)}\right)$$

of being accepted. If the candidate is accepted, $X_{t+1} = Y$, and if it is not accepted, $X_{t+1} = X_t$, indicating that the state remains unchanged. Using this scheme to simulate X_t, $t = 1, 2, 3 \ldots$, and given that the candidate distribution $q(\cdot|X)$ fulfills certain conditions, it can be proven that the process will have $\pi(\cdot)$ as its stationary distribution [7]. This means that for large t, X_t will approximate a random variable from the distribution $\pi(\cdot)$.

2.3.2 Simulated annealing

The simulated annealing method is closely related to the Metropolis-Hastings algorithm and is often used in optimization [19]. In such problems, the state $x \in S$ that minimizes the objective function $\Phi(x)$ is sought, where S is the set of all possible or legal states. The idea is to model the target distribution $\pi(\cdot)$ in the Metropolis-Hastings algorithm after $\Phi(\cdot)$ such that states yielding low values on the objective function are more likely to appear. However, unlike the normal Metropolis-Hastings simulation, the target distribution will be dependent on a temperature parameter T. This parameter will be allowed to vary throughout the simulation and the target distribution will have the form $\pi_T(\cdot)$.

The method is called simulated annealing as it aims to simulate the thermodynamic process of cooling solids. At high temperatures, all particles of a solid will be arranged randomly, but as the temperature decreases, the particles will arrange themselves to low energy states. During the cooling process, if it proceeds slowly, the cooling solid will reach thermal equilibrium at each temperature T, meaning that the probability that the solid will be in a state x with energy E_x will follow the Boltzmann distribution,

$$P(x) = \frac{1}{N_T} \exp\left(\frac{-E_x}{k_B T}\right)$$

where N_T is a normalization constant that is a summation over all possible macroscopic states and k_B is the Boltzmann constant. In an optimization problem, the objective function can be used as energy levels of the various states in S and the annealing process can be simulated with the Metropolis-Hastings algorithm by using the Boltzmann distribution as the target distribution. By lowering the temperature slowly enough, thermal equilibrium is maintained at each temperature, and as the temperature approaches zero, only the states with the lowest energy state will have a non-zero probability of appearing. These low energy states can thus be used as approximate solutions to the optimization problem. Viewing the simulated annealing algorithm in the perspective of the Metropolis-Hastings algorithm, it can be thought of as running a series of simulations with different temperatures. Each simulation will be run at a temperature lower than the previous one and use the final state of the previous simulation as its initial state.

Using the Boltzmann distribution as the target distribution causes the Metropolis criterion to become

$$\alpha(X, Y) = min\left(1, \exp\left(\frac{\Phi(Y) - \Phi(X)}{T}\right)\right)$$

assuming a symmetric candidate distribution, that is $q(X|Y) = q(Y|X)$ for all X, Y.

2.3.3 Cooling scheme

The way the temperature is varied in the simulated annealing algorithm is called the cooling scheme, which needs to be carefully selected as it affects the convergence [19]. The two main families of cooling schemes are homogenous and inhomogenous simulated annealing. In homogenous simulated annealing, the algorithm performs several iterations at each temperature, allowing it to reach thermodynamic equilibrium. If the algorithm uses the temperatures t_i, $i = 1, 2, 3, \ldots$, and an infinite number of iterations are made at each temperature, van Laarhoven and Aarts proved that the algorithm will converge to a global optimum provided $\lim_{i\to\infty} t_i = 0$ [19].

Inhomogenous simulated annealing means that the temperature is decreased at each step according to a predefined scheme. The inhomogenous algorithm also converges asymptotically to a global optimum given that the temperature scheme t_k goes to zero at a rate slower than $O\left(log(k)^{-1}\right)$. However, using such a cooling scheme would be impractical because of how slowly the temperature decreases and would in theory amount to a random search in state space [20]. The rate of temperature decrease is crucial to avoid getting stuck in local optima.

2.3.4 Candidate distribution

In order for the Metropolis-Hastings and simulated annealing algorithms to work effectively and converge within a reasonable timespan, the choice of candidate distribution is important [7]. For numeric reasons, it is also important that it is computationally expedient to sample the distribution. A common way to choose a distribution is to, for each state x, define a set N_x as its neighborhood and then define the candidate distribution as

$$q(y|x) = \begin{cases} \frac{1}{|N_x|} & \text{if} y \in N_x \\ 0 & \text{if} y \notin N_x \end{cases}$$

which amounts to choosing a candidate from the current state's neighborhood randomly.

2.4 Simplified protein models

Modelling protein folding using the methods presented in section 2.3 requires constructing a model that simulates the behavior of the amino acid residues. This behavior includes both the interaction between the amino acid residues as well as their spatial location. The model should strike a balance between fidelity to the biological theory and the constraints imposed by computational resources. Too simple a model limits the scope of information in the simulation while a model that is too complex risks placing undue stress upon the computations. Additionally, models of exceeding complexity may impede expedient interpretation of results [21].

2.4.1 Hydrophobic-polar model

The hydrophobic-polar protein folding model, oftentimes shortened to the HP model, is a highly simplified model based upon the view that the strongest force in protein folding is hydrophobic interactions between residues, in keeping with the hydrophobic collapse theory mentioned in section 2.2. All amino acids are thus divided into two broad categories depending upon their hydrophobicity: hydrophobic (nonpolar) or polar [22].

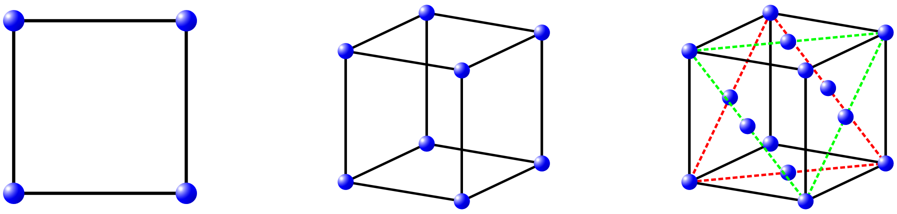
In order to promote protein folding, the HP model assigns scores to the interactions between hydrophobic (H) and polar (P) residues. The HH interaction is given a favorable score of -1 while the other interactions, HP and PP, are either ignored or given a positive score. As the thermodynamic hypothesis states that the native conformation occurs at a global energy minimum, finding the correct conformation should be equivalent to finding the configuration that yields the lowest score.

Due to the simplistic nature of the HP model, it is unable to successfully mimic the many intricacies of protein folding. However, this same simplicity proves to be one of its greatest strengths as it allows for ease of implementation and analysis of results. Since its formulation by Dill in 1985, it has been used in multiple studies [23, 24, 25, 26], further acknowledging its usefulness.

2.4.2 Lattice models

Computational representations of protein folding often seek to decrease the scale of the problem. A simplification often employed involves limiting which spatial positions amino acid residues can adopt by implementing a discrete lattice [27].

Three simple lattice models are shown in figure 2.3. As each free point on the lattice can only be occupied by one amino acid residue at a time, the protein can be thought of as carrying out a self-avoiding walk when finding its conformation [27].



(a) Square. **(b)** Cubic. **(c)** Face-centered cubic.

Figure 2.3: One unit cell for three different lattice models: square, cubic, and face-centered cubic (FCC).

Implementing the Metropolis-Hastings algorithm on the different lattice models changes the dynamics of the algorithm as the acceptable next conformations and the affecting neighbors vary as a result of the current state. Different models yield different characteristics, and consequently different results.

Square and cubic lattice

The most elementary representation of protein conformation is the square lattice model seen in figure 2.3a. It consists of orthogonal, discrete points, each having an equal distance to all of its neighbors. One of the model's great strengths is its simplicity as it allows for rapid analysis, both analytical and intuitive, and thus serves as a springboard in understanding the protein folding problem [28]. This simplicity, however, is a significant drawback as well since the model diverges decidedly from the packing displayed by real-world proteins.

As proteins occupy three-dimensional space, another dimension must be added to make a more representative lattice structure. By stacking square lattices directly atop one another, a three-dimensional, cubic lattice is formed. Each discrete point, corresponding to a vertex in figure 2.3b, is an equal distance away from all six neighbors. These neighbors, represented by dotted circles in figure 2.4, are the only residues that may interact with the given amino acid.

The major drawback of using square and cubic lattices is the fact that they are unable to circumvent the parity problem, illustrated for the two-dimensional case

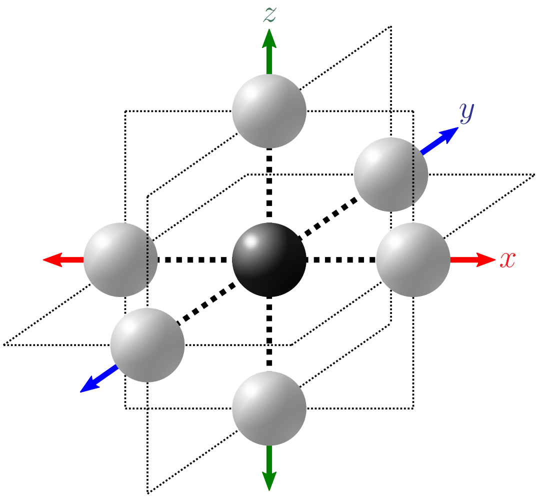


Figure 2.4: Potential neighbors for an amino acid on a cubic lattice.

in figure 2.5. On these lattices, it is impossible for two consecutive odd or even residues to be neighbors, eliminating their ability to interact with one another. As such, a sequence with large stretches of alternating H and P residues will not be able to fold on these lattices [28].

Face-centered cubic lattice

As the parity problem does not occur in nature, a more refined lattice model is necessary in order to allow more realistic interactions. One such model is the face-centered cubic lattice, shortened to FCC. It is similar to the cubic lattice except for the addition of lattice points at the center of each side of the cube. When these amended cubic structures are organized into a grid, the FCC system arises.

When using the FCC lattice, the dynamics of the system are changed as the number of neighbors doubles compared to the cubic lattice. In the FCC system, a lattice point in the corner of the cube shown in figure 2.3c will have twelve neighbors, thereby increasing the degrees of freedom. When the initial residue is at the corner point, only those residues found in face-centered positions are regarded as neighbors as the adjacent corners are too far away to interact. Thus, the residue has four neighbors in the xy plane, four in the xz plane, and four in the yz plane, as shown in figure 2.6.

The parity problem is overcome in the FCC lattice by the inclusion of the face-centered lattice positions. The layers in the FCC system are offset one another, meaning that the four neighbors that lie above a given amino acid residue are the

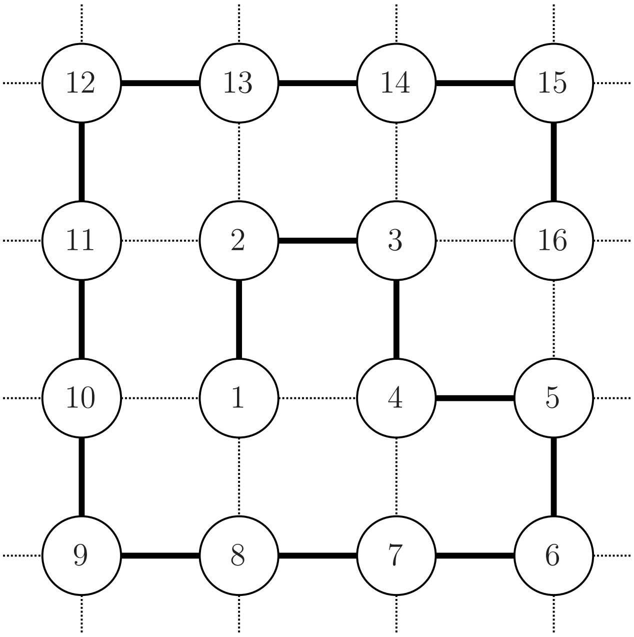
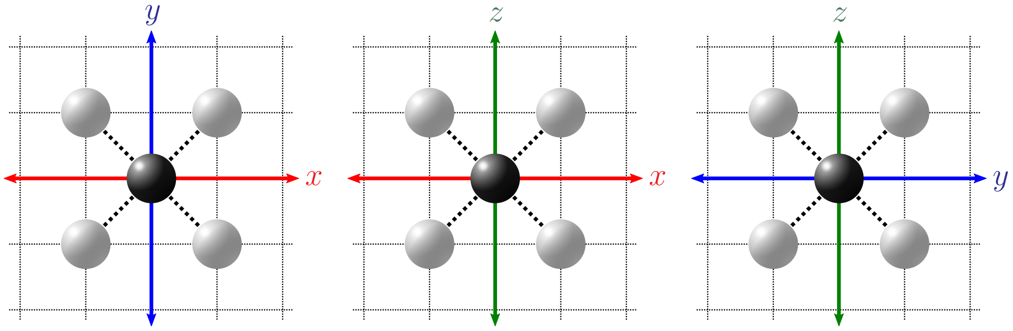


Figure 2.5: Illustration of the parity problem. The odd (even) residues can not interact with other odd (even) residues. This means that a sequence with alternating H and P residues is unlikely to fold correctly.



(a) xy plane. **(b)** xz plane. **(c)** yz plane.

Figure 2.6: Potential neighbors for an amino acid residue on a FCC lattice.

same distance away as those in the same plane. A conformation such as that in figure 2.7 is thus possible, allowing for, in this case, odd-numbered residues to interact with one another.

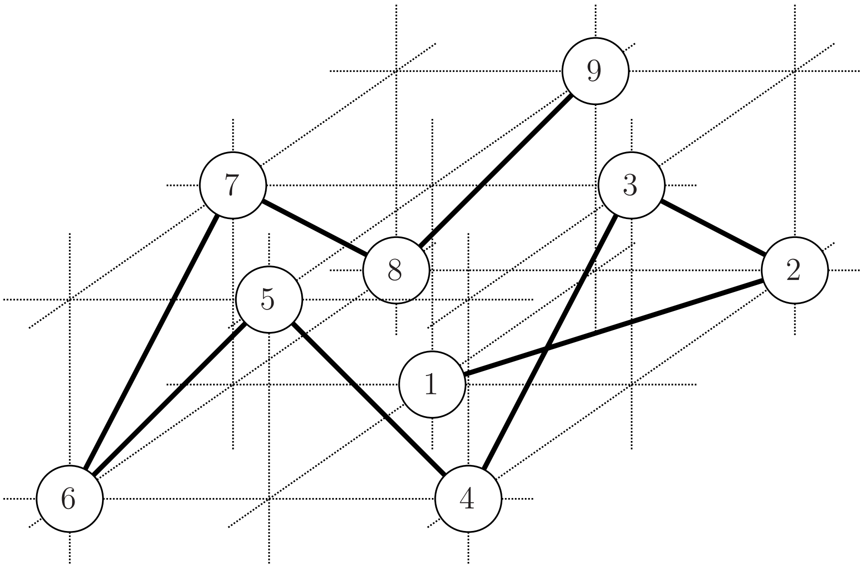


Figure 2.7: Illustration of how the parity problem is overcome on the FCC lattice. The intraresidual distance between neighbors is equal to the diagonal of the cubic face. As such, the odd (even) residues can now interact with other odd (even) residues.

Packing density

According to Rose and Wolfenden, an amino acid chain tends to adopt as compact a structure as possible when folding into a protein [3]. This tight packing must accommodate two competing forces: steric hindrance and the desire to minimize empty space [29]. Steric hindrance arises due to the fact that two amino acids cannot occupy the same spatial position. To overcome this issue, the amino acid side chains of the hydrophobic core fit together in an almost jigsaw-like pattern, thereby leaving minimal empty space.

The packing density, p_d, serves as a measurement of the degree of compactness a specific model displays [29]. This is calculated by dividing the total volume occupied by the molecules, in this case spherical amino acid residues, by the total volume of the unit cell.

$$p_d = \frac{V_{\text{residues}}}{V_{\text{unit cell}}}$$

Using this equation, it becomes apparent that, under the assumption that all amino acid residues are of the same size, the FCC model displays a higher degree of packing than the cubic model, as illustrated by the following calculations:

$$p_{d_{\text{cubic}}} = \frac{\frac{4}{3}\pi r^3}{(2r)^3} = \frac{4\sqrt{2}\pi}{3} \approx 0.5236$$
$$p_{d_{\text{FCC}}} = \frac{4\frac{4}{3}\pi r^3}{(2r\sqrt{2})^3} = \frac{\pi}{3\sqrt{2}} \approx 0.7405$$

2.4.3 Off-lattice model

In nature, proteins are not bounded by a discrete lattice model. As such, modelling them without a lattice results in a closer representation of reality. The greatest drawback of discarding the lattice framework is the added cost in the form of greater computational complexity. This additional cost makes modelling a completely realistic off-lattice protein infeasible, requiring certain concessions to be made. The major simplification of the implemented off-lattice model is the same as that in the lattice models, namely the inclusion of the HP model, albeit a slightly modified version.

Extended HP model

The off-lattice model employed in this thesis uses an adapted version of the HP model, which includes the addition of a third residue type called N for neutral. This residue is used to model the polypeptide backbone of the protein and as such does not interact with the other two residues. Instead, the H and P residues are redefined as side chains of the amino acid, meaning that each such residue occupies a position adjacent to an N.

Structure

To further increase the similarity to actual proteins, certain restrictions where placed upon which spatial arrangements the residues can adopt. The N residues were made into a chain by connecting them with a rigid bond of unit length. Due to the geometry displayed by proteins in vivo, the angle of the chain connecting any three successive neutral residues is fixed at 120°. Additionally, the side chain residues are connected to the N residues by bonds of unit length, where the direction of said bond is made to coincide with the direction of the cross product formed by the two adjacent NN bonds. The limits imposed by this structural geometry are summarized in figure 2.8.

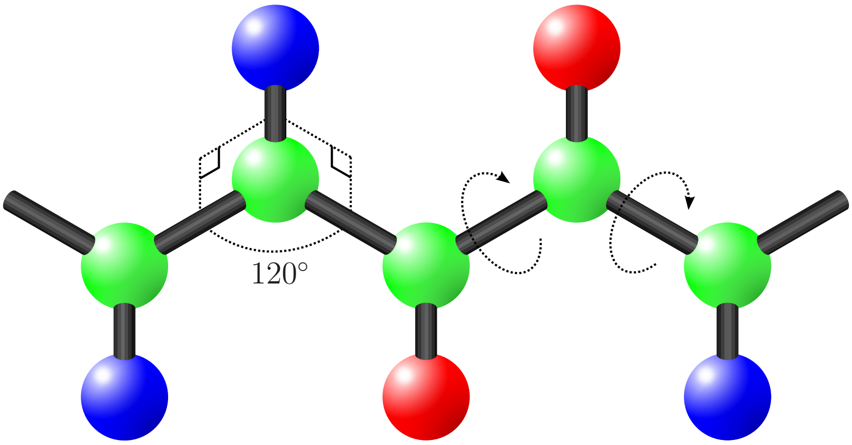


Figure 2.8: Diagram explaining the geometry of the off-lattice model. The angle between two adjacent NN bonds is set to 120° and each amino acid side chain is orthogonal to the N residue it is corrected to. New conformations are adopted through rotating about the bonds.

2.4.4 Energy function

In the lattice models, two H residues are considered neighbors if they occupy two adjacent lattice points. As all neighboring pairs are equidistant, the implementation of a scoring function is merely a matter of granting a favorable, energetically negative score to each hydrophobic interaction.

As the off-lattice model allows for varying distances between interacting residues, the energy function employed by the lattice models, mentioned in 2.4.1, is inadequate. Adjusting it to fit the framework of the off-lattice model necessitates compensating for the fluctuation in distance between neighboring residues. This is achieved through utilization of the Lennard-Jones potential [30]

$$V_{LJ} = \varepsilon \left[\left(\frac{r_m}{r} \right)^{12} - 2 \left(\frac{r_m}{r} \right)^{6} \right]$$

where the depth of the potential well is denoted ε, the distance between the two residues is given by r, and the distance at which the lowest potential $-\varepsilon$ is reached is represented by r_m.

In keeping with the principles of the HP model, only HH interactions are given a favorable energy score. The depth of the potential well is set to 1 and the distance at which this potential is obtained, r_m, is set to a unit distance of 1. As in the lattice models, interactions between any other combination of residues is given a score of 0 and the total energy is found by summing the potentials.

2.4.5 Search neighborhoods

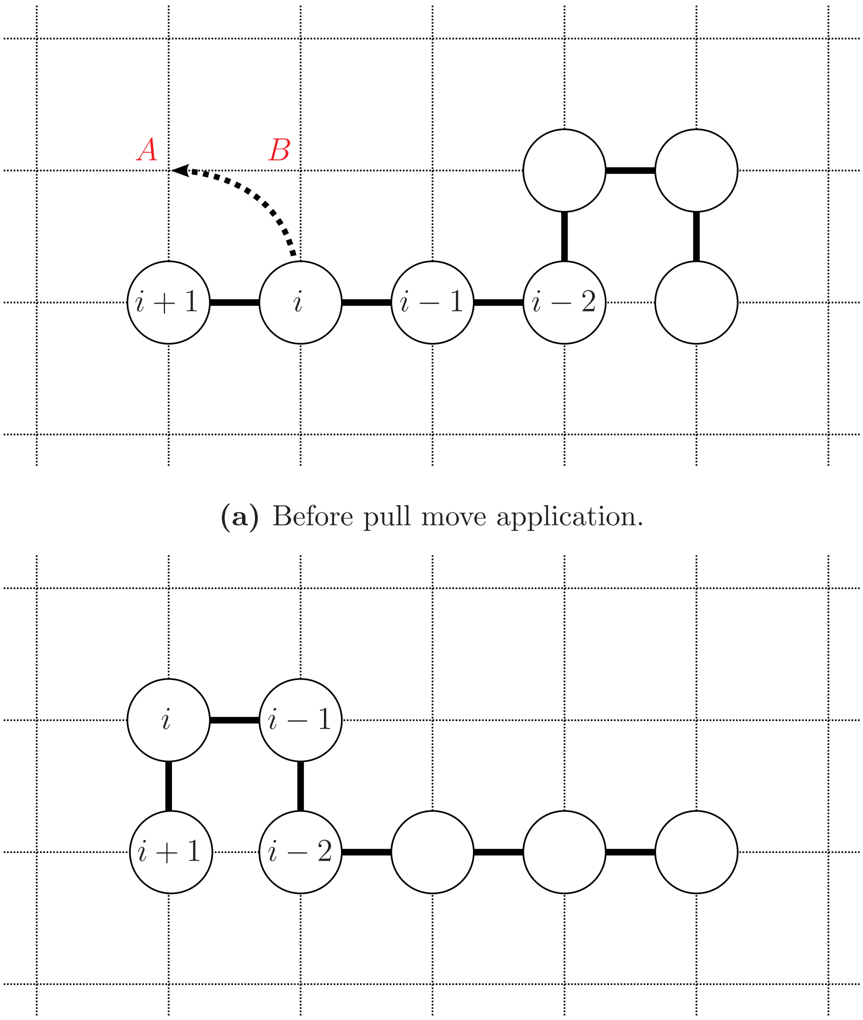
Since the candidate functions used in Metropolis-Hastings algorithm often are based on neighborhoods, it is necessary to construct a set of rules outlining which states these are composed of. In the protein folding problem, the neighborhood of a conformation includes proteins that display the same geometry as the current state with the addition of one slight perturbation or *move*.

Lattice search neighborhoods

There are various ways of constructing these neighboring conformations for lattice models. In the nascence of protein folding, single residue moves such as *single residue end* and *corner moves* were dominant [31]. Later, these moves were combined with the *two residue crankshaft move* into a single conformational neighborhood in the work of Gurler et al. [32].

Another, more recently developed method of generating neighboring conformations called *pull moves* is used in this study. This method was first applied to the protein folding problem by Lesh, Mitzenmacher, and Whitesides in 2003 [33] as a way to improve the existing methods. It has since been shown to have substantially better performance than earlier methods [24].

To understand how the move works, see figure 2.9. Suppose that during a step in the algorithm that, for some residue i, there is an empty position A neighboring residue $i+1$. Suppose also that there is a position B which neighbors both residue i and the empty position A. The pull move is initiated by moving residue i to position A. Then, if it is not already there, residue $i-1$ is moved to position B. The chain is then checked for validity. If the chain is free of gaps, the pull move is considered completed. Otherwise, as position B is the neighbor to the now empty position of residue i, the residue $i-2$ is moved to fill the space previously occupied by i. The chain is once again checked for validity and, if it is invalid, the residue $i-3$ is moved to the previous position of residue $i-1$. This continues until either a valid chain is constructed or the chain terminates. The search neighboorhood



(a) Before pull move application.

(b) After pull move application.

Figure 2.9: Schematic diagram of pull moves. Residue i is seeking to go to position A. In doing so it pulls residue $i-1$ with it up to position B. The position previously occupied by i is filled by $i-2$ and so on until a conformation with no gaps is obtained.

for the protein chain can thus be defined as all possible conformations obtainable through the application of a single pull move.

Off-lattice search neighborhood

The framework of the off-lattice model does not allow for conformational changes similar to those of the lattice models. However, the positional constraints imposed by the model structure decrease the degrees of freedom that have to be taken into account when searching for neighboring conformations. The only modifications that can be made to an existing state that will not result in the breaking of bonds is rotation about the bonds connecting two adjacent N residues, illustrated by the curved arrows in figure 2.8. The off-lattice model's search neighborhood is thus comprised of all such rotations that do not result in steric overlap.

2.5 Analyzing conformations

In order to detect patterns emerging in the results, a means of measuring similarity is introduced. These similarity measurements may then be used along with statistical methods to identify and analyze clusters of similar conformations.

2.5.1 k-medoids clustering

One method of partitioning a set of data points into clusters of similar data is the k-medoids partitioning method [34]. This method aims to divide the data set into k clusters using solely the definition of the distance or dissimilarity function $d(i, j)$, which is to be interpreted as the dissimilarity between the data points i and j. Each cluster C resulting from one particular clustering will contain one data point i which minimizes the function $\sum_{j \in C} d(i, j)$. This data point i is called the medoid of the cluster.

The quality of clustering is measured by the mean distance between data points and the medoids of their respective clusters, where one seeks to find as compact clusters as possible. To find high quality clusters, the Partitioning Around Medoids (PAM) method as described by Kaufman and Rousseeuw can be used [35].

To visualize a clustering and get a sense of its quality, a silhouette plot can be made [35]. The silhouette plot is based on the notion of a silhouette coefficient $s(i)$ for each data point i. This coefficient is a measure of similarity between how similar a given data point is to the other data points in its own cluster and how similar it is to data points of the cluster second closest to it.

The value of the coefficient can range from -1 to 1. If the majority of data points in a cluster have a silhouette coefficient close to 1, that cluster's structure is regarded as clear and well-defined. Coefficients closer to zero indicate that the cluster structure is weak and that it may be arbitrary or artificial, while negative coefficients indicate that the clustering is wrong and the data points might better fit in another cluster.

2.5.2 Root-mean-square deviation

The root-mean-square deviation (RMSD) is a method of calculating similarity between two conformations of a chain [36]. It is defined as

$$d(i,j) = \sum_{k=1}^{n} (x_k^i - R_{ij} x_k^j)^2,$$

where x_k^i is the coordinate of residue k in the i:th chain, translated such that the centroid of the chain is at the origin and R_{ij} is a rotation reflection matrix such that $d(i,j)$ is minimized. To calculate the matrix R_{ij}, the algorithm described by Kabsch can be used[37].

RMSD is commonly used to assess if a simulated protein fold is successful by calculating the root-mean-square deviation of the predicted conformation and the empirically observed native conformation [38, 36]. Maiorov and Crippen showed that the root-mean-square deviation of the folds of two proteins is correlated with the similarity of the amino acid sequences of the proteins[38]. If the amino acid sequences have few residues in common, the folds tend to have a higher root-mean-square distribution.

3

Software

THE IMPLEMENTATION IS DIVIDED into three separate parts: simulation, visualization, and evaluation. The first one runs the algorithms and the protein folding simulation, the second displays a protein chain in its current state, and the last portion consists of functions and tools to evaluate the results.

The details of the underlying theory and mathematics that is implemented is explained in section 2.3. Only specific design choices and special procedures will be assessed in this section of the thesis.

3.1 Simulation software

The main program of this thesis was implemented in Haskell, which was chosen as a compromise between ease of development and performance. Haskell's strong type system coupled with the clever optimizations made by The Glasgow Haskell Compiler (GHC) makes it simple to write fast, reliable code.

The implementation of the main simulation program is modular and is composed of different, separate parts. The algorithmic framework that implements the theory, presented in the sections 2.3.1 and 2.3.2, is independent of the model chosen and is used in all simulations presented in this thesis. Among the different models, the off-lattice model is implemented separately while the different on-lattice models share a substantial amount of code.

3.1.1 Optimization framework

While implementing the algorithmic framework, extra focus was placed upon generality and extensibility. The simulated annealing algorithm was implemented such that it could operate on general candidate generating and energy functions. These functions were also allowed to operate on general state data structures, which meant that it was easy to implement several protein models, both lattice and off-lattice, within the simulated annealing framework. The code implementing different lattice types, energy, and kinetic models was written independently of the implementation of the main algorithm.

In the implementation, the cooling scheme is specified explicitly by a list containing the temperature at each step. After each step, only the latest state is retained and all previous states will be discarded. As mentioned in section 2.3.1, these are not needed anymore as the Markov property dictates that the next state depends solely upon the current one.

The Metropolis-Hastings algorithm normally takes into account the probability of generating a candidate given the current state, $q(X|Y)$, and the probability of generating the current state given the candidate, $q(Y|X)$, when determining the probability of accepting the candidate. This is not done in this implementation as no feasible way to compute these values for the models used was found. A simplified version of the core implementation of the simulated annealing algorithm can be seen in figure 3.1.

```
metropolisHastings scorefunc candfunc initial temps =
    foldM mhStep inititial temps
    where
        mhStep state t = do
            cand <- candfunc state
            u <- random (0.0, 1.0)
            if u <= exp ((scorefunc candstate -
                              scorefunc state)/t)
                then return cand
                else return state
```

Figure 3.1: A simplified excerpt from the implementation of the simulated annealing algorithm.

3.1.2 Lattice models

For the lattice models, implementation of candidate generating and energy functions was focused on utilizing pull moves and the HP model in a way that would allow using the same code for a variety of lattice types. The protein states were modelled as lists of coordinates in a lattice system, where each coordinate represents the position of one residue. The code implementing the pull move and HP model operates on general coordinate data structures using only a predefined set of operations. This means that by implementing these operations, it is easy to introduce a new lattice model without having to reimplement the pull move and HP model.

The pull move only requires two functions for each lattice type. The functions needed are one that can determine whether two coordinate positions are considered neighboring and one that, given two neighboring coordinates c_1 and c_2, is able to enumerate all pairs of coordinates c_A and c_B such that c_A neighbors c_B, c_1 neighbors c_A, and c_2 neighbors c_B (see the description of position A and B in section 2.4.5). The HP model only requires a function enumerating all neighbors of a given coordinate.

3.1.3 Off-lattice model

The residues in the off-lattice model were modelled as solid spheres with a diameter of 1 unit, and lists of coordinates and bond vectors were used to represent the state of the protein.

Functions able to rotate residues about arbitrary axes were implemented, as well as functions able to decide whether a rotation is valid or not; that is, whether the residues overlap each other in any intermediary stage of the rotation. Rudimentary collision detection was implemented in order to achieve this in a reasonably effective manner.

Collision detection was performed in two phases: a broad phase and a narrow phase. The broad phase was used to quickly determine which objects have a possibility of colliding, and those in danger of colliding are flagged accordingly. The narrow phase is slower and more exact, and determines whether the pairs of objects flagged in the broad phase actually collide.

The candidate generating function consisted of choosing a bond about which to rotate as well as the amount to rotate at random. If the chosen rotation results in a valid rotation, the result of the rotation is chosen as the candidate. When this is not the case, another bond and angle is chosen at random until a valid rotation is found.

The collision detection framework was used in the implementation of the scoring function. The Lennard-Jones potential used had a cut-off distance of 2.5 units, that is to say that a distance greater than 2.5 units results in a potential that is virtually 0. This means that the interaction volume of the residues could be modeled as spheres and that the collision detection framework mentioned above could be used to determine which residues interact. This of course saved considerable computing time since the $O(n^2)$ energy calculations that would otherwise be required could in most cases be reduced.

3.2 Graphical user interface

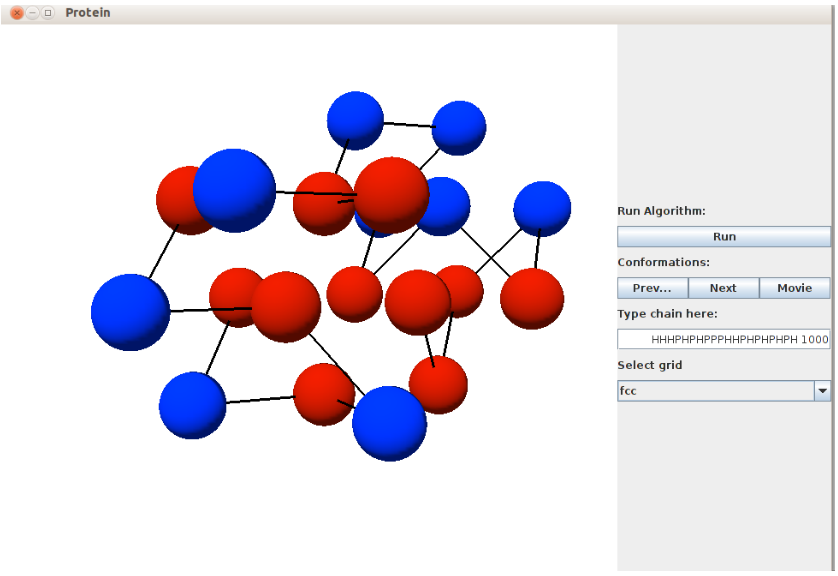
The graphical user interface (GUI) used to visualize the output from the simulation was written in Java. This GUI was created with the use of OpenGL (open graphics library) combined with Java's standard libraries. Windows, buttons, and text were implemented using the libraries *awt* and *swing*, while the drawing of the folded protein and camera were implemented with JOGL, a wrapper library that extend OpenGL features to the Java environment [39].

The GUI is a simple program with some basic features that run the simulations, toggle between conformations, select different lattice models, and rotate the view. A priority was to make the program user-friendly but powerful and, by hiding complex options in a command field, the front end of the program was kept simple.

3.2.1 Model

It is possible to run the simulation software from the GUI, a screenshot of which can be seen in figure 3.2. The user selects a lattice type to run the simulation on, inserts the desired chain of hydrophobic and polar residues and the number of iterations desired in the appropriate field.

The simulation is done in Haskell, and the result is piped back to the GUI. The GUI takes the resulting conformations as input, and for each residue in every conformation the coordinates are stored together with the residue type. The stored conformations are accessible through the navigation pane, which allows the option to step through an entire evaluation. The first residue in every conformation is



Type chain here: HHHPHPHPPPHHPHPHPHPH 1000

Select grid: fcc

Figure 3.2: Screenshot of the GUI used in this thesis. The sequence of hydrophobic and polar residues and the number of iterations to be run are used as input and the lattice type is chosen through a drop-down list.

always placed at the origin, since the protein tends to otherwise drift away when the number of iterations increase.

The rendering is done with an animator that continually displays the desired conformation. Material and lighting properties in JOGL are utilized in order to distinguish between the hydrophobic (red) and polar (blue) residues. Furthermore, by using the residues coordinates, rods are drawn between two neighbouring residues, showing linkage.

3.2.2 Camera

To get a spatial comprehension of the rendered protein, a so called *trackball camera* was implemented. An trackball camera can be thought of as a sphere with a viewpoint on its surface. The focus point of the camera is at the trackball's origin, and changing the focus point will change where the trackball is located. An example of how the camera works can be seen in figure 3.3.

The camera has the option to move along the surface of the trackball, which is achieved by pressing the left mouse button while dragging the figure. Holding the right mouse button down while dragging will offset the origin of the trackball by an equivalent distance. Zooming is done by changing the trackball's radius

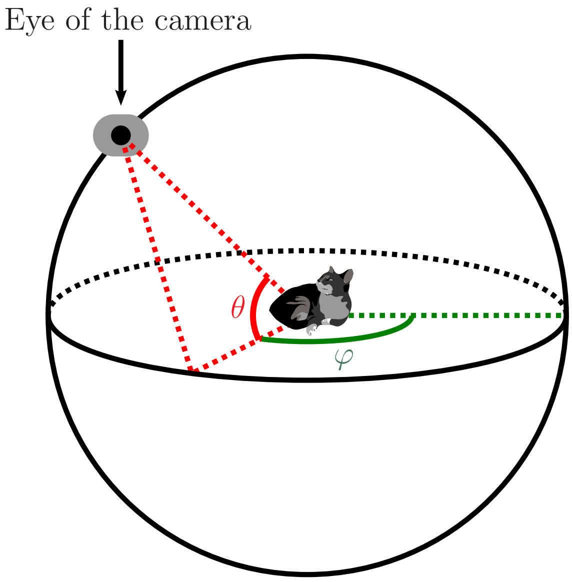


Figure 3.3: An example of an trackball camera with a cat as the focus point. ϕ is the longitude and θ is the latitude.

via scrolling the mouse wheel. A small crosshair, whose size is dependent upon the distance from the origin of the camera, indicates where the camera is being focused, making it easier for the user to avoid getting lost.

4

Results

The data obtained by implementing the simulated annealing algorithm on lattice and off-lattice models as well as statistical analysis is presented herein. Specific focus has been placed upon displaying how well the algorithm finds low energy structures, patterns found in the solutions, and the amount of time necessary to run the simulation. As a frame of reference, the results from the implemention of the algorithm on the lattice models are compared to results from HPstruct, an effective tool based upon constraint programming [10].

4.1 Benchmark sequences

The benchmark sequences seen in table 4.2 were chosen from an article by Thachuk, Shmygelska, and Hoos [24]. These sequences have been thoroughly studied [40, 41, 42, 43] and proven to result in global energy minima. Each sequence is 48 residues long with unique patterns of Hs and Ps, shown in 4.1.

Table 4.1: Benchmark sequences [24].

ID	Length	Protein sequence
S2-2	48	$H_4PH_2PH_5P_2HP_2H_2P_2HP_6HP_2HP_3HP_2H_2P_2H_3PH$
S2-3	48	$PHPH_2PH_6P_2HPHP_2HPH_2(PH)_2P_3H(P_2H_2)_2P_2HPHP_2HP$
S2-4	48	$PHPH_2P_2HPH_3P_2H_2PH_2P_3H_5P_2HPH_2(PH)_2P_4HP_2(HP)_2$
S2-5	48	$P_2HP_3HPH_4P_2H_4PH_2PH_3P_2(HP)_2HP_2HP_6H_2PH_2PH$
S2-6	48	$H_3P_3H_2PH(PH_2)_3PHP_7HPHP_2HP_3HP_2H_6PH$
S2-7	48	$PHP_4HPH_3PHPH_4PH_2PH_2P_3HPHP_3H_3(P_2H_2)_2P_3H$
S2-8	48	$PH_2PH_3PH_4P_2H_3P_6HPH_2P_2H_2PHP_3H_2(PH)_2PH_2P_3$
S2-9	48	$(PH)_2P_4(HP)_2HP_2HPH_6P_2H_3PHP_2HPH_2P_2HPH_3P_4H$

4.2 Energy comparison

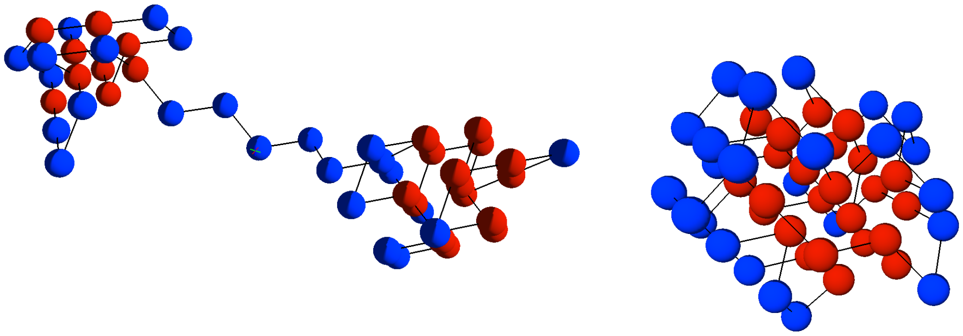
The cubic and FCC lattice models were simulated using the HP energy model with a fast cooling of $50,000$ iterations and a slow cooling of $100,000$ iterations. The temperature was decreased quadratically and each chain was simulated 100 times for the slow cooling scheme and $1,000$ times for the fast cooling scheme. The variance in the number of simulations was due to time constraints, though it was ascertained that 100 simulations was adequate to obtain the necessary results. The minimum energies obtained from these simulations are presented in table 4.2.

Table 4.2: The lowest energies obtained when simulating the benchmark sequences 100 times for each cooling scheme and model. All of the simulations for the slow cooling scheme and the first 100 for the fast cooling scheme were analyzed. The entries marked with an asterisk (*) are the best values when comparing the fast and slow cooling schemes.

	HPstruct		Fast		Slow	
ID	Cubic	FCC	Cubic	FCC	Cubic	FCC
S2-2	−34	−69	−32*	−52	−31	−68*
S2-3	−34	−72	−32	−63	−32	−69*
S2-4	−33	−71	−30	−67	−31*	−68*
S2-5	−32	−70	−30	−66	−31*	−70*
S2-6	−32	−70	−31	−52	−32*	−68*
S2-7	−32	−70	−30	−64	−31*	−68*
S2-8	−31	−69	−29	−53	−29	−67*
S2-9	−34	−71	−32	−69	−33*	−69

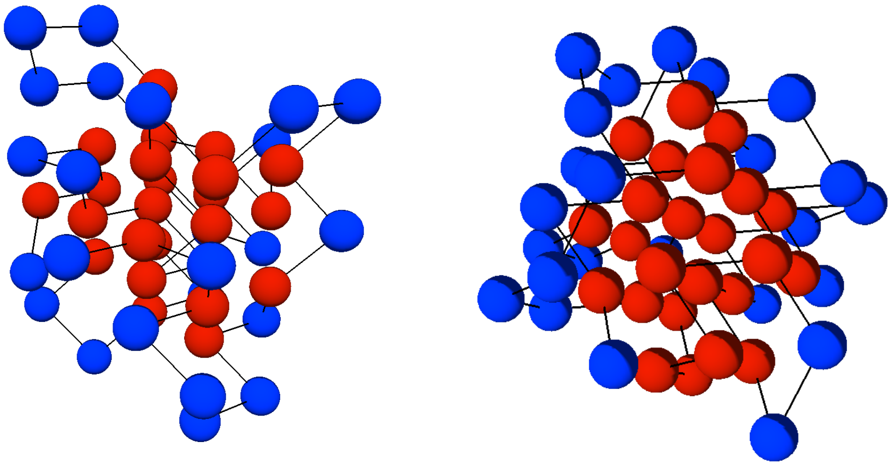
4.3 Sample chains

As seen in table 4.2, the energy values obtained upon simulation differed from those found by HPstruct with regard to the FCC lattice. The cubic lattice, on the other hand, showed similar values for both the simulation and HPstruct. As sequence S2-2 and sequence S2-9 were structures with a bad respectively good energy score compared to HPstruct, these chains were studied in further detail. Figure 4.1 and 4.2 show the lowest energy conformations of structure S2-2 respectively S2-9 found by both the simulation and HPstruct.



(a) Simulated structure. **(b)** HPstruct structure.

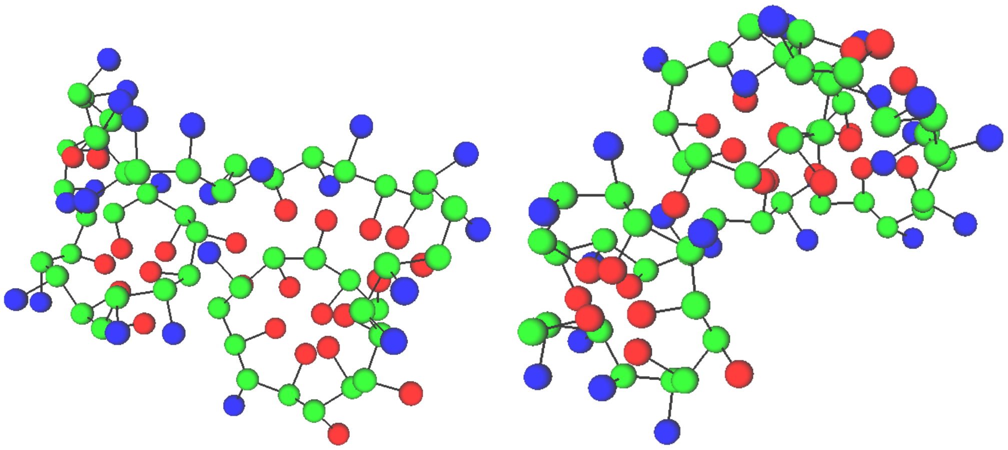
Figure 4.1: Visual representation of S2-2 on FCC using simulated and HPstruct results. The red and blue sphere represent hydrophobic (nonpolar) and polar amino acid residues respectively.



(a) Simulated structure. **(b)** HPstruct structure.

Figure 4.2: Visual representation of S2-9 on FCC using simulated and HPstruct results. The red and blue sphere represent hydrophobic (nonpolar) and polar amino acid residues respectively.

To see how well the off-lattice performed, two conformations are seen in figure 4.3. They are presented without any comparing results due to the fact that no suitable off-lattice model was found as a comparison.



(a) Chain S2-2. **(b)** Chain S2-9.

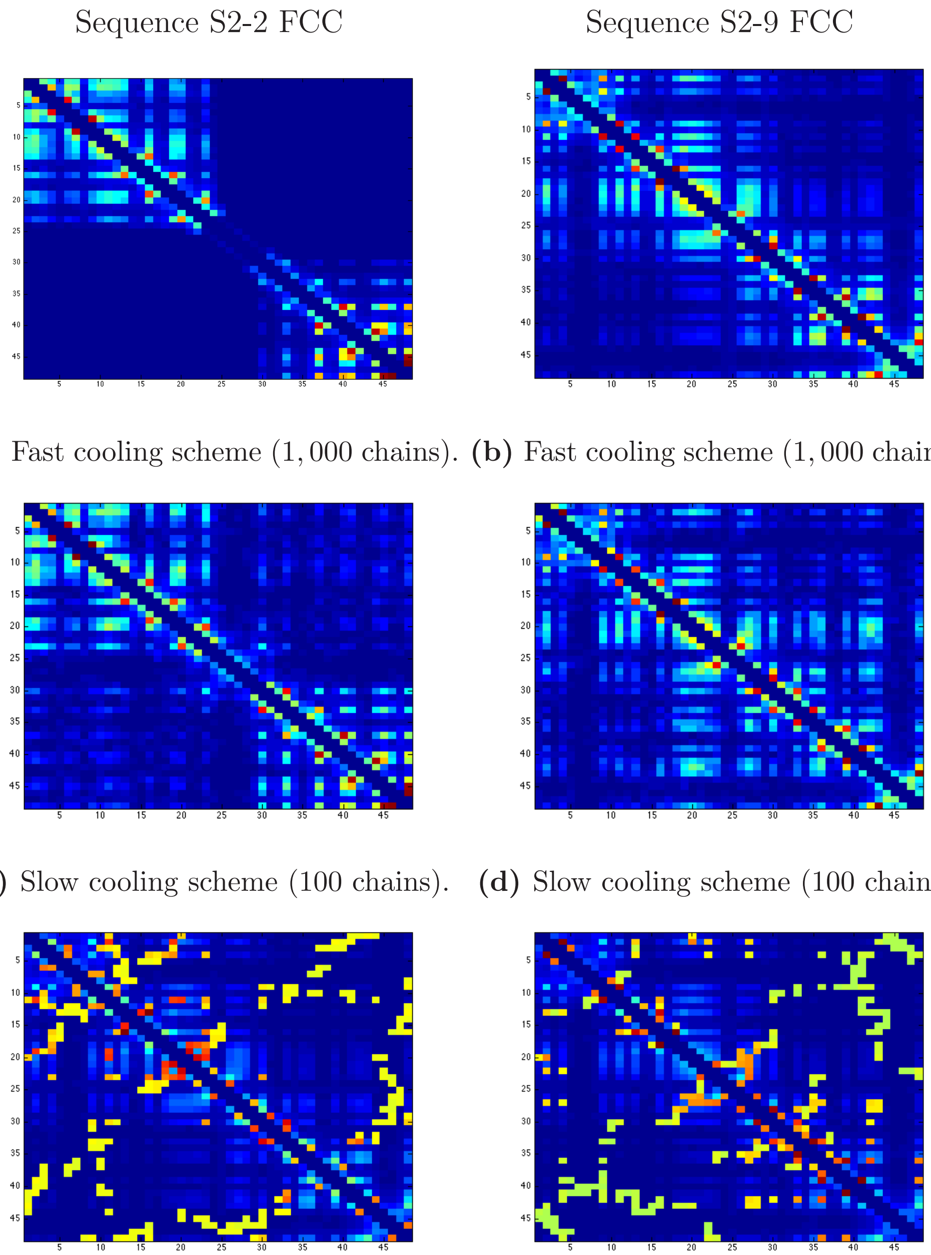
Figure 4.3: Algorithm implemented on the off-lattice model, run with a slow cooling scheme. The neutral backbone mentioned in section 2.4.3 is colored green and the red and blue spheres represent hydrophobic (nonpolar) and polar amino acid side chains respectively.

4.3.1 Connection matrices

For a further investigation into how the connections are distributed in the chain, connection matrices plotted as heat maps have been used. Aggregated values of all simulations are used and hot colors represent connections often found in our chains, while cool colors represent an absence of connections. The color of the pixel at position (i, j) corresponds to the prevalence of connection between the i:th and j:th residue in the chain.

It is important to note that the resulting heat map shows an average of all of the results. In the case of the simulation, this includes every result obtained, not only those displaying minimal energy. In stark contrast, HPstruct only averages those results displaying the global minimum energy. This is due to the fact that the program only allows the user to extract structures that display optimal energy.

Figures 4.4 and 4.5 show the connection matrices for S2-2 and S2-9, both for the simulation and for HPstruct.

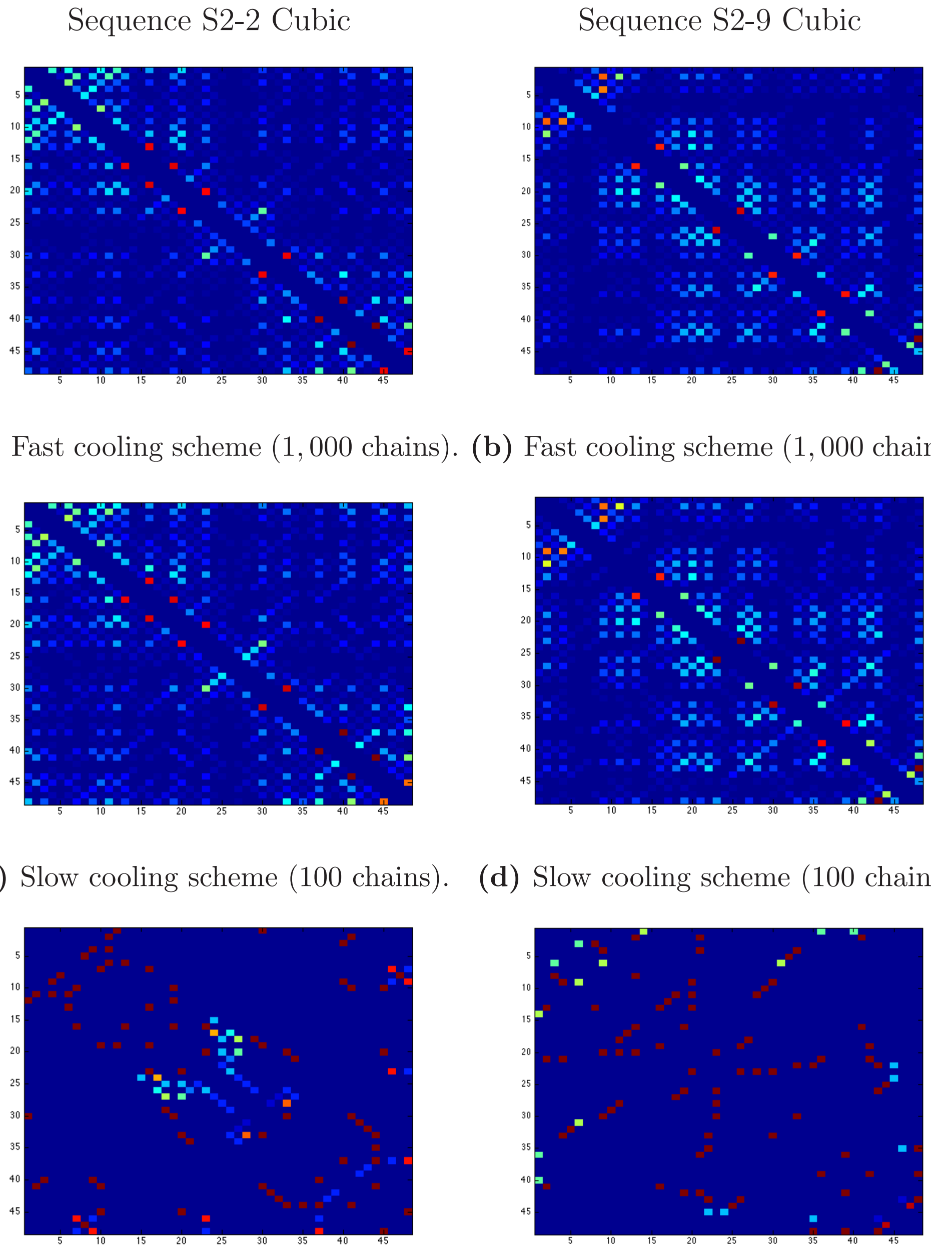


(a) Fast cooling scheme (1, 000 chains). **(b)** Fast cooling scheme (1, 000 chains).

(c) Slow cooling scheme (100 chains). **(d)** Slow cooling scheme (100 chains).

(e) HPstruct results (13 chains). **(f)** HPstruct results (13 chains).

Figure 4.4: Aggregation of the most common connections between residues for sequences S2-2 and S2-9 on the FCC lattice. The HPstruct results are an aggregation of 13 different optimal energy structures. Two types of simulations are shown, one using a fast cooling scheme at 50, 000 iterations and one slow at 100, 000 iterations. The number on the axes represents the residue position in the chain.



Sequence S2-2 Cubic

Sequence S2-9 Cubic

(a) Fast cooling scheme (1, 000 chains).

(b) Fast cooling scheme (1, 000 chains).

(c) Slow cooling scheme (100 chains).

(d) Slow cooling scheme (100 chains).

(e) HPstruct results (13 chains).

(f) HPstruct results (13 chains).

Figure 4.5: Aggregation of the most common connections between residues for sequences S2-2 and S2-9 on the cubic lattice. The HPstruct results are an aggregation of 13 different optimal energy structures. Two types of simulations are shown, one using a fast cooling scheme at 50, 000 iterations and one slow at 100, 000 iterations. The number on the axes represents the residue position in the chain.

4.3.2 Similarities of results

To investigate if it is possible to see any common structure in folds from the same sequence compared to folds from other sequence, a cluster analysis was carried out. The results were analyzed by a k-medoids clustering assuming two clusters using PAM. The result of this analysis is the silhouette plot, explained in section 2.5.1, seen in figure 4.6.

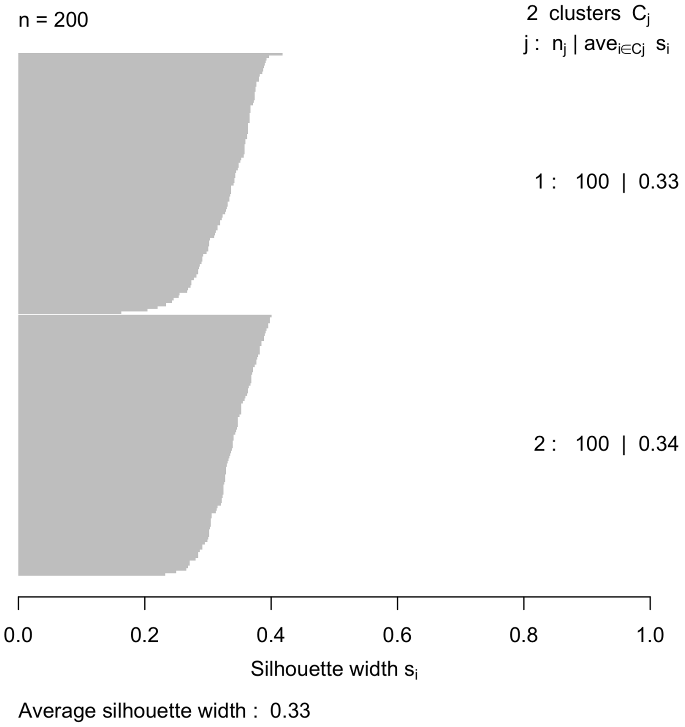


Figure 4.6: Silhouette plot of k-medoid clustering of hundred results each from S2-2 and S2-9 using RMSD as distance function.

The clustering succeeded in clustering all folds such that all conformations from the same chain clustered together repeatedly. The average silhouette coefficient was 0.33 and 0.34 for the two clusters respectively. The average distance between a data point and the medoid of the whole data set was measured to be 19.4, while the average distance from a data point to the medoid of its cluster was measured to be 15.1 and 11.3 respectively. In figure 4.7, a multidimentional scaling from the metric defined by RMS distance onto two-dimensional Euclidian space of the data is shown. The maximum error in one of the scaled distances is 5.44 and the mean error is 0.0685.

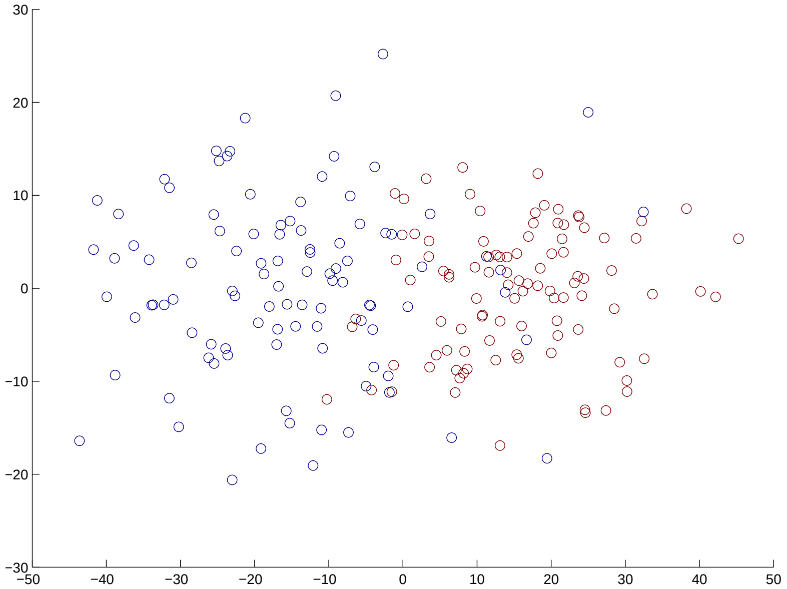


Figure 4.7: Multidimentional Scaling plot of the RMS distances between conformations of S2-2 (blue) and S2-9 (red).

4.4 Run time

Theory dictates that different lattice models should differ with respect to the amount of time necessary to fold a chain of residues. In order to visualize the difference, speed runs on the models mentioned in section 2.4.2 were conducted. Residue sequences of lengths 12, 24, 36, 48, 64, 90, and 135 were studied and the results are presented in figure 4.8. As suspected, the off-lattice model took the longest amount of time, followed by FCC, then cubic.

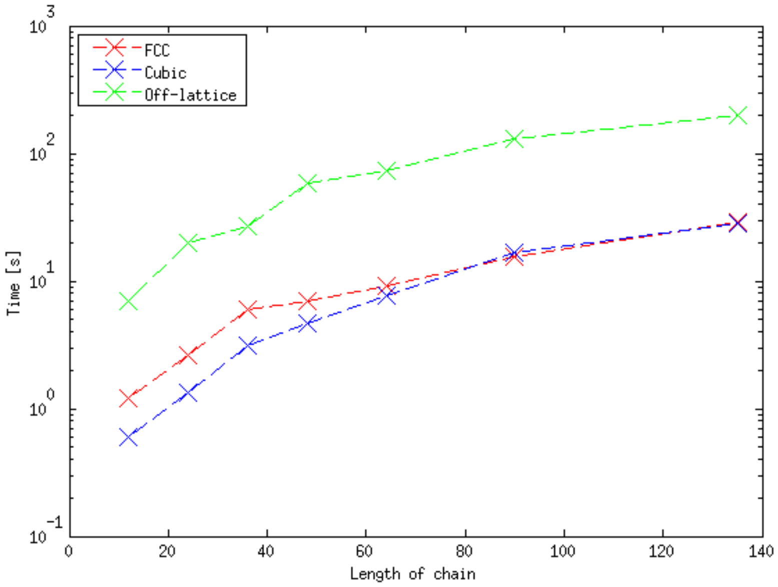


Figure 4.8: The runtime of the simulated annealing algorithm as the chain length increases.

5

Discussion

Due to the open-ended nature of the protein folding problem, assessment of the degree of success and the reliability of the results is a difficult endeavor. This difficulty is compounded by the multitude of different approaches used in similar studies, few of which overlap. Comparing the results of two different studies is oftentimes futile as each research effort results in different models, algorithms, and scoring functions. It is not an easy task to evaluate if a folded chain is a success or a failure, as the question of what makes a successful fold is not fully answered, though a low energy state is thought to be indicative of a correct protein fold in accordance with the thermodynamic hypothesis.

5.1 Energy score

The result of the different cooling schemes in table 4.2 show that, with regards to the cubic lattice, both cooling schemes perform well compared to HPstruct. However, the slow cooling scheme is marginally better than the fast on the cubic lattice. This is in accordance with the theory explained in 2.3.2. If the temperature is decreased slowly enough, the probability distribution for every temperature will be close to thermal equilibrium, yielding better performance.

Regarding the results on the FCC lattice, the slow cooling scheme outperformed the fast. The slow cooling scheme produced better results on every chain, except chain S2-9, where it was the same. When considering the energy landscape, the topography gets more complicated with the FCC lattice, as the search neighborhood is greater. Consequently, having a more complex model increases the time until thermal equilibrium. This is due to the fact that the Boltzmann distribution depends on a factor N_T, a summation of all possible macroscopic states. This might explain why the fast cooling scheme underperformed on an FCC lattice but not the cubic lattice.

As far as the energy score is concerned, the algorithm performed well in comparison to HPstruct, with the occasional outlying result using the fast cooling scheme on the FCC lattice. Switching to a slow cooling scheme yields a better result, but this increases the time of simulation as it requires more computational resources. While this is not a significant issue for the chains tested, increasing the length of the sequences could potentially make the implementation of slow cooling schemes infeasible.

Being unable to compare the energies of the off-lattice model, a potential theory would be that the time required to reach thermal equilibrium would be even greater, and therefore requiring even longer execution time.

5.2 Heat map

The different cooling schemes for chains S2-2 and S2-9 were compared to HPstruct. Important to note is that since only 13 optimal results are used from HPstruct, any conclusions drawn from this comparison should be regarded with a certain degree of skepticism. That being said, it is possible to see distinct patterns in the different heat maps, evident in both the comparison of the same chain on different algorithms and between different chains.

5.2.1 Face-centered cubic lattice

When comparing the results of chain S2-2 to HPstruct, seen in figure 4.4, the pattern implies that the simulated approach has a large number of local connections while the best structures of HPstruct has a greater degree of long-range interaction. The heat map showing the results of the HPstruct algorithm also shows that there is a certain pattern of connections in the top right and bottom left corner which are missing in the heat maps from simulated annealing. This may suggest that there is a certain set of energetically favorable connections that the simulated annealing simulation is not able to find consistently.

The sequence of chain S2-2 (see table 4.1) has six consecutive polar residues in the center. The simulated annealing heat maps suggest that it is unlikely that the hydrophobic residues from both ends combine and create a single core. Instead, the long polar portion of the sequence separates the hydrophobic residues and one core forms in each end, as is seen by the abundance of connections in the top left and bottom right corners of the heat maps. These two ways of folding are visualized in figure 4.1. A possible reason to why the more energetically favorable long-range interactions seen in the HPStruct simulations are not reached by the simulated annealing simulation could be that these optimal structures have an energy barrier which is too large to overcome. This is supported by the fact that it is possible to discern that long-range interactions are slightly more prevalent in the slow cooling scheme.

Chain S2-9 performs well on both fast and slow cooling scheme, but the slow cooling scheme produces chains with generally larger amount of long-range interactions. Looking at the heat map of HPstruct, it is evident that the terminal ends have connections to one another, but this behavior is only displayed to a marginal degree when using simulated annealing. As the long-range interactions observed in the pattern of HPstruct appears when using both the slow and fast cooling schemes, it is reasonable to believe that the energy barrier to reach these conformations is lower. This might help explain why the fast cooling scheme performs well on this chain.

5.2.2 Cubic lattice

As in the FCC case, the heat maps of chain S2-2 (figure 4.5a and 4.5c) are lighter towards the top left and bottom right corner, meaning that the local interactions at the ends are slightly more preferred. The heat maps of S2-9 (figure 4.5b and 4.5d) are concentrated along the diagonal, meaning that overall local interactions are preferred.

The almost evenly spread heat map on both lattices means that the conformations of HPstruct are reachable. However the lack of strong patterns also imply that no consistent convergence is occurring, implying that the folding is quite random.

5.3 Cluster analysis

It is clear that there is some significant structural difference between folds of S2-2 and S2-9 as measured by the RMSD. It was not possible to discern any clusters within the results from the same chains, and thus it can not be said that the solutions are divided into families or are centered around a limited set of solutions. The solutions seem well distributed over some part of the state space, suggesting that the possible solutions obtained from the algorithm are numerous. However, there is no way to know if the solutions are distributed evenly among the part of state space yielding the energy levels obtained. It is apparent from figure 4.7 that the intraresidual distances cluster by residue sequence. However, it is not possible to know for sure whether this clustering is due to the method itself or to intrisic properties of the problem.

5.4 Native state

Drawing conclusions about how well the simulations approximate the native state is difficult, primarily due to the simplifications made to accommodate the limits placed by computational resources. The HP model, while effective, can only approximate the energetics of amino acid interaction. The potential of the native state, meanwhile, may depend upon nuanced potential differences that this approximation is unable to imitate [43]. It is thus conceivable that the benchmark sequences chosen do not converge to a single native state and instead display several energy minima, none of which is necessarily the actual native state.

As mentioned in section 4.1, multiple studies using the same benchmark sequences showed a tendency to converge to an energy minimum, assumed to be the native state. This premise may be faulty as it is possible that different yet similar arrangements yield the same energy, making the exact identification of which is the native state an impossibility.

5.5 Run time

The time required to run the simulations of the different protein models is shown in figure 4.8 in section 4.4. Conclusions may be drawn from the results, even if it is difficult to extract any exact measurements of the computational complexity for the various protein models. For instance, it is apparent that the time taken to run the FCC and cubic lattice models grows at the same rate, even though it would appear that the FCC lattice model has a greater overhead run time cost. The similar growth rate is expected considering that both models are simulated using the same framework.

The complexity of the off-lattice model appears to be an order of magnitude greater than that of the lattice models. This is not a surprising result considering the additional computations required by such a model. The additional complexity of the off-lattice model is one of the major reasons for difficulties in acquiring sufficient data for the statistical analysis. The complexity of computation meant that the trouble faced when running off-lattice simulations will escalate as the length of the chains grows.

5.6 Off-lattice

Collecting and analyzing data for the off-lattice implementation posed a number of difficulties, the first and foremost of which is the difficulty of comparing data gathered from the off-lattice simulations with any of the other results. The second difficulty is that of actually collecting data, as the increase in computational complexity from the on-lattice models was greater than first expected. Once it was completed, there was insufficient time left to run simulations and collect the data.

While the data gathered may be insufficient for the rigors of statistical analysis, interesting conclusions can be drawn based upon visual inspection of the resulting folds. As this subtype of off-lattice model has yet to be thoroughly examined, we are pleased that we have been able to implement such a novel model and that it appears to work. The conformations at which the simulation arrives, even if the energies vary, seems to have the H residues clustered together forming a hydrophobic core. This in turn meant that most of the simulations resulted in a compact chain.

5.7 Comparing the models

When comparing the results across the different lattices, it was apparent that the simulated annealing algorithm generally obtained better results when used on the cubic lattice than it did on the FCC lattice. This is to say that the deviation between the simulation and HPstruct results was significantly less for the cubic lattice, as seen in table 4.2.

Looking at the heat maps of the cubic and FCC lattice, it appears that the former tended to converge to a global energy minimum while the latter did not. The cubic lattice had connections that were spread along the whole chain, implying a mix of local and long-range interactions between residues. The simulated annealing on the cubic lattice showed more consistent energy scores and heat maps than simulated annealing on the FCC lattice. However, using the FCC lattice meant that the search and solution space increased, making it naturally harder to find the best solution. This difficulty was compounded when using the simulated annealing algorithm on the off-lattice model, as it in turn has an even larger search and solution space.

There is strong indication that if the algorithm is run long enough with a sufficiently slow cooling scheme, a protein with low energy and desirable conformation will eventually be found. Since the simulated annealing algorithm appears to work on both lattice and off-lattice model, it was not apparent that a particular model was inferior to the others, meaning that none of the models could be discarded. When comparing the results across the different models, one must weigh time complexity against performance, but also bear in mind the relevance of the different cubic, FCC and off-lattice models.

6

Conclusion

During the course of this project, we have studied the problem of protein folding using a simulated annealing algorithm on cubic, FCC, and off-lattice models, with the underlying purpose of gaining insight into the problem of protein folding. The implemented method for simulating the folding process on these simplified models is an effective means of finding low energy conformations. Additionally, by using statistical cluster methods it was possible to verify that there are patterns in the solutions, although it is unclear whether these patterns arise due to the simulation process or if they are intrinsic to the chains themselves. In order to determine the cause of these patterns, other methods, such as genetic algorithms [25], ant colony optimization [40] or replica exchange Monte Carlo algorithms [24], may be explored and their results compared.

In analyzing our results, we mainly used two methods. Our GUI allowed us to visualize the conformation adopted by a protein after simulation, allowing us to quickly see whether or not our result seemed feasible. To get an all-encompassing view of patterns displayed over several simulations, we constructed a connection heat map that visualized which amino acid residues interacted most often.

Through simulating protein folding with our algorithm, we were able to explore the effect the cooling scheme has upon overcoming energy barriers when seeking a low energy conformation. A significant energy barrier that could not be overcome by a fast cooling scheme was the formation of long-range interactions. Despite this barrier, the proteins showed a propensity towards forming hydrophobic cores, in keeping with the hydrophobic collapse model.

Perhaps one of the most exciting portions of this thesis was the implementation of a novel off-lattice model. Due to the added complexity of the theory and limited time frame, the off-lattice model has much room for improvement. Some potential improvements could include refining the performance and accuracy of the algorithm, increasing the number of affecting parameters, abandoning the HP model, and considering the quaternary structure as well as the properties of the surrounding liquids. If developed further, we believe that this model has the potential to become ubiquitous in future studies of simplified protein folding.

The open-ended nature of the protein folding problem means that we have but scratched the surface in this thesis. Much can be done to improve the methods implemented in order to obtain better results and thereby a deeper understanding of this puzzle.

References

[1] C. B. Anfinsen. “Principles that Govern the Folding of Protein Chains.” In: *Science* 181.4096 (July 1973), pp. 223–230. DOI: science.181.4096.223.

[2] K. A. Dill and H. S. Chan. “From Levinthal to pathways to funnels.” In: *Nature Structural Biology* 4.1 (Jan. 1997), pp. 10–19. DOI: 10.1038/nsb0197-10.

[3] G. D. Rose and R. Wolfenden. “Hydrogen Bonding, Hydrophobicity, Packing, and Protein Folding.” In: *Annual Review of Biophysics and Biomolecular Structure* 22 (June 1993), pp. 381–415. DOI: 10.1146/annurev.bb.22.060193.002121.

[4] K. A. Dill, S. Bromberg, K. Yue, K. M. Fiebig, D. P. Yee, P. D. Thomas, and H. S. Chan. “Principles of protein folding – A perspective from simple exact models.” In: *Protein Science* 4.4 (Apr. 1995), pp. 561–602. DOI: 10.1002/pro.5560040401.

[5] M. Gruebele. “Protein folding.” In: *Methods* 52.1 (Sept. 2010), pp. 1–2. DOI: 10.1016/j.ymeth.2010.08.010.

[6] U. H. E. Hansmann and Y. Okamoto. “New Monte Carlo algorithms for protein folding.” In: *Current Opinion in Structural Biology* 9.2 (Apr. 1999), pp. 177–183. DOI: 10.1016/S0959-440X(99)80025-6.

[7] S. Chib and E. Greenberg. “Understanding the Metropolis-Hastings Algorithm.” In: *The American Statistician* 49.4 (Nov. 1995), pp. 327–335. DOI: 10.2307/2684568.

[8] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell.* 5th ed. Garland Science, 2008. ISBN: 978-0-8153-4105-5.

[9] M. T. Madigan, J. M. Martinko, D. A. Stahl, and D. P. Clark. *Brock Biology of Microorganisms.* 13th ed. Benjamin Cummings, 2012. ISBN: 978-0-321-64963-8.

[10] M. Mann, C. Smith, M. Rabbath, M. Edwards, S. Will, and R. Backofen. "CPSP-web-tools: a server for 3D lattice protein studies." In: *Bioinformatics* 25.5 (Mar. 2009), pp. 676–677. DOI: `10.1093/bioinformatics/btp034`.

[11] C. L. Brooks III, M. Fruebele, J. N. Onuchic, and P. G. Wolynes. "Chemical physics of protein folding." In: *Proceedings of the National Academy of Sciences of the United States of America* 95.19 (Sept. 1998), pp. 11037–11038. DOI: `10.1073/pnas.95.19.11037`.

[12] F. H. C. Crick. "On Protein Synthesis." In: *Symposia of the Society for Experimental Biology* 12 (1958), pp. 138–163.

[13] E. Buxbaum. *Fundamentals of Protein Structure and Function*. Springer, 2007. ISBN: 978-0-387-68480-2.

[14] H. Frauenfelder. *The Physics of Proteins*. Ed. by S. S. Chan and W. S. Chan. Springer, 2010. DOI: `10.1007/978-1-4419-1044-8`.

[15] P. Atkins and L. Jones. *Chemical Principles. The Quest for Insight.* 4th ed. New York, NY: W.H. Freeman and Company, 2008. ISBN: 978-0-7167-7355-9.

[16] S. Govindarajan and R. A. Goldstein. "On the thermodynamic hypothesis of protein folding." In: *Proceedings of the National Academy of Sciences of the United States of America* 95.10 (May 1998), pp. 5545–5549. DOI: `10.1073/pnas.95.10.5545`.

[17] G. D. Rose and J. E. Dworkin. "The Hydrophobicity Profile." In: *Prediction of Protein Structure and The Principles of Protein Conformation*. Ed. by G. D. Fasman. New York: Plenum Press, 1989. Chap. 15, pp. 625–646. ISBN: 978-1-4612-8860-2.

[18] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. "Introducing Markov chain Monte Carlo." In: *Markov Chain Monte Carlo in Practice*. Ed. by W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. Chapman & Hall, 1996. Chap. 1, pp. 1–19. DOI: `10.1007/978-1-4899-4485-6`.

[19] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Ed. by M. Hazewinkel. D. Reidel Publishing Company, 1987. DOI: `10.1007/978-94-015-7744-1`.

[20] Y. Nourani and B. Andresen. "A comparison of simulated annealing cooling strategies." In: *Journal of Physics A: Mathematical and General* 31.41 (Oct. 1998), pp. 8373–8385. DOI: `10.1088/0305-4470/31/41/011`.

[21] A. Kolinski and J. Skolnick. "Reduced models of proteins and their applications." In: *Polymer* 45.2 (Jan. 2004), pp. 511–524. DOI: `10.1016/j.polymer.2003.10.064`.

[22] K. A. Dill. "Theory for the Folding and Stability of Globular Proteins." In: *Biochemistry* 24 (1985), pp. 1501–1509. DOI: `10.1021/bi00327a032`.

[23] D. K. Klimov and D. Thirumalai. "Criterion that Determines the Foldabity of Proteins." In: *Physical Review Letters* 76.21 (May 1996), pp. 4070–4073. DOI: 10.1103/PhysRevLett.76.4070.

[24] C. Thachuk, A. Shmygelska, and H. H. Hoos. "A replica exchange Monte Carlo algorithm for protein folding in the HP model." In: *BMC Bioinformatics* 8 (Sept. 2007), p. 342. DOI: 10.1186/1471-2105-8-342.

[25] J.-J. Tsay and S.-C. Su. "An effective evolutionary algorithm for protein folding on 3D FCC HP model by lattice rotation and generalized move sets." In: *Proteome Science* 11 (2013), S19. DOI: 10.1186/1477-5956-11-S1-S19.

[26] D. L. Shaw, A. S. M. S. Islam, M. S. Rahman, and M. Hasan. "Protein folding in HP model on hexagonal lattices with diagonals." In: *BMC Bioinformatics* 15 (Jan. 2014), S7. DOI: 10.1186/1471-2105-15-S2-S7.

[27] A. Bechini. "On The Characterization and Software Implementation of General Protein Lattice Models." In: *PLoS One* 8.3 (Mar. 2013), e59504. DOI: 10.1371/journal.pone.0059504.

[28] S. E. Decatur. *Protein Folding in the Generalized Hydrophobic-Polar Model on the Triangular Lattice*. Tech. rep. MIT-LCS-TM-559. Accessed: 2014-05-17. Massachusetts Institute of Technology, May 1996. URL: http://bitsavers.trailing-edge.com/pdf/mit/lcs/tm/MIT-LCS-TM-559.pdf.

[29] F. M. Richards and W. A. Lim. "An analysis of packing in the protein folding problem." In: *Quarterly Reviews of Biophysics* 26.4 (Nov. 1994), pp. 423–498. DOI: 10.1017/S0033583500002845.

[30] J. Daintith, ed. *A Dictionary of Chemistry*. 6th ed. Oxford University Press, 2008. ISBN: 978-0-19-920463-2.

[31] P. H. Verdier and W. H. Stockmayer. "Monte Carlo Calculations on the Dynamics of Polymers in Dilute Solution." In: *The Journal of Chemical Physics* 36.1 (Jan. 1962), pp. 227–235. DOI: doi:10.1063/1.1732301.

[32] M. T. Gurler, C. C. Crabb, D. M. Dahlin, and J. Kovac. "Effect of Bead Movement Rules on the Relaxation of Cubic Lattice Models of Polymer Chains." In: *Macromolecules* 16.3 (Mar. 1983), pp. 398–403. DOI: 10.1021/ma00237a012.

[33] N. Lesh, M. Mitzenmacher, and S. Whitesides. "A Complete and Effective Move Set for Simplified Protein Folding." In: *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology* (2003), pp. 188–195. DOI: 10.1145/640075.640099.

[34] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001. DOI: 10.1007/978-0-387-21606-5.

[35] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Vol. 344. John Wiley & Sons, 2005. DOI: `10.1002/9780470316801`.

[36] F. E. Cohen and M. J. E. Sternberg. "On the Prediction of Protein Structure: The Significance of the Root-mean-square Deviation." In: *Journal of Molecular Biology* 138.2 (Apr. 1980), pp. 321–333. DOI: `10.1016/0022-2836(80)90289-2`.

[37] W. Kabsch. "A solution for the best rotation to relate two sets of vectors." In: *Acta Crystallographica A* 32 (Sept. 1976), pp. 922–923. DOI: `10.1107/S0567739476001873`.

[38] V. N. Maiorov and G. M. Crippen. "Significance of Root-Mean-Square Deviation in Comparing Three-dimensional Structures of Globular Proteins." In: *Journal of Molecular Biology* 235.2 (Jan. 1994), pp. 625–634. DOI: `10.1006/jmbi.1994.1017`.

[39] *JogAmp*. Accessed: 2014-04-01. URL: `http://jogamp.org/`.

[40] A. Shmygelska and H. H. Hoos. "An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem." In: *BMC Bioinformatics* 6 (Feb. 2005), p. 30. DOI: `10.1186/1471-2105-6-30`.

[41] U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler. "Testing a New Monte Carlo Algorithm for Protein Folding." In: *Proteins: Structure, Function, and Bioinformatics* 32.1 (July 1998), pp. 52–66.

[42] T. C. Beutler and K. A. Dill. "A fast conformational search strategy for finding low energy structures of model proteins." In: *Protein Science* 5.10 (Oct. 1996), pp. 2037–2043. DOI: `10.1002/pro.5560051010`.

[43] K. Yue, K. M. Fiebig, P. D. Thomas, H. S. Chan, E. I. Shakhnovich, and K. A. Dill. "A test of lattice proten folding algorithms." In: *Proceedings of the National Academy of Sciences of the United States of America* 92.1 (Jan. 1995), pp. 325–329. DOI: `10.1073/pnas.92.1.325`.