LEBANESE AMERICAN UNIVERSITY

B

Scatter Search for Protein Structure Prediction

Rt
00569
c.1

by

Christine Houry Kehyayan

B.S., Computer Science, Lebanese American University, 2004

Thesis submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

Division of Computer Science and Mathematics

February 2008

LEBANESE AMERICAN UNIVERSITY

Thesis approval Form

Student Name:	Christine Houry Kehyayan	I.D.: 200100522
Thesis Title :	Scatter Search for Protein Structure Prediction	
Program :	M.S. in Computer Science	
Division/Dept :	Computer Science and Mathematics	
School :	Arts and Sciences - Beirut	

Approved/Signed by:

Thesis Advisor	Dr. Nashaat Mansour	[redacted]
Member	Dr. Faisal AbuKhzam	
Member	Dr. Hassan Khachfe	[redacted]

Date: February 4, 2008

Plagiarism Policy Compliance Statement

I certify that I have read and understood LAU's Plagiarism Policy. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.

This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Christine Houry Kehyayan

Signature: [redacted] Date: February 4, 2008

Acknowledgment

I would like to thank my advisor Dr. Nashaat Mansour for his guidance throughout my Thesis work. A thanks is also to Dr. Faisal AbuKhzam and Dr. Hassan Khachfe for being on my thesis committee.

I would like to express my sincere gratitude to the Lebanese American University whose financial support during my graduate studies made it all possible.

Finally, I would like to thank my family and friends for their long support.

Abstract

Proteins are organic compounds made up of chains of amino acids. These amino acids are formed from atoms and the chain fold into complex 3-dimensional structures based on their chemical and physical properties. A protein is characterized by its 3D structure, which defines its biological function. The protein structure prediction problem has real-world significance where several diseases such as Alzheimer, cystic fibrosis, mad cow disease, and many cancers are associated with the wrong folding of proteins. Computational methods for predicting protein structures have recently gained popularity. In this thesis, we present a scatter search algorithm for predicting 3D structures of proteins. Given the protein's sequence of amino acids and data collected from known protein structures, our algorithm produces a 3D structure that aims to minimize the energy function associated with protein folding. Scatter search is an evolutionary approach that is based on a population of solution candidates. These candidates undergo evolutionary operations that combine search intensification and diversification over a number of iterations. We evaluate our algorithm on three proteins taken from a protein data bank. The results show that our algorithm is able to produce 3D structures with good sub-optimal energy values. Also, the root mean square deviations of these structures from the reference proteins are promising within limits imposed by the assumptions used.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Proteins are made up of combinations of twenty different amino acids. These amino acids are formed from atoms, namely H, C, N, O, and S, with two or more C's are bonded to one another, thus, proteins are organic compounds of different types and roles in living organisms. Structural proteins are responsible for maintaining the structure of some biological components, or generating mechanical forces to ease the motility process of organisms. Enzymes are proteins that act as catalysts to fasten chemical reactions in living systems. Other proteins have different functions, such as, signal carrying, oxygen transport, and antibody defense. Initially a protein is a linear chain of amino acids, ranging - in general - from 50 up to 5,000 amino acids. These amino acids fold into complex structures based on the chemical and the physical properties of the atoms. These folds determine the function of the protein provided that the folding is in its native structure, which is the correct three-dimensional structure of the protein. As a result, any misfold of the protein leads to failure in protein accomplishing its function (Setubal and Meidanis, 1997). Prusiner (1998) showed that some unrelated diseases such as alzheimer, cystic fibrosis, mad cow disease, and many cancers are associated with the wrong folding of proteins. However, not all protein structures can be determined because of the limitations in the biophysical techniques used. This gives the motivation to further study the protein structure prediction (PSP) problem, which is to computationally generate three-dimensional native or native-like structures of target proteins given amino acid sequences.

1.1 Computational Approaches to 3D Protein Structure Prediction

Computational approaches for PSP fall into two groups. Approaches in the first group use predicted protein structures from protein data banks (PDB). These approaches assume proteins evolve with time, thus, they belong to families of already predicted proteins. Approaches in the second group solely rely on the given amino acid sequence without obtaining any structural data from ancestors. There are two main approaches in the first group: comparative modeling and fold recognition. In the second group, there are many approaches that are all called *ab initio* methods. This section gives an overview of some of the approaches of the two groups.

1.1.1 Comparative Modeling

Comparative modeling is also known as sequence alignment or homology modeling. It uses sequences of known structures in PDB to align with the given or the target protein's sequence for which the 3D structure is to be predicted. This approach assumes that new proteins are generated as a result of evolutions that mutate the sequences by adding, deleting, or substituting some amino acids, and these mutations do not change the structures, i.e., similar sequences end up having similar structures. The accuracy of the alignment process here is important, since it largely contributes to the accuracy of the prediction. For 50% sequence similarity between the target sequence and any sequence in protein databases, comparative modeling gives results as good as experimental methods (Kopp and Schwede, 2004). For 30-50% sequence similarity most of the Cα-atoms are expected to have 3.5 Angstroms (Å), 10^{-10} meters, deviation from their real positions (Kopp and Schwede, 2004), this indicates a moderate fit between the target sequence and the reference sequence. For cases less than 30% sequence similarity the prediction is expected to have large errors (Kopp and Schwede, 2004). Comparative modeling consists of five steps: (1) finding one or more suitable structural templates for the target sequence from families or classes of known structures in databases; (2) aligning the target sequence with the template sequence(s); (3) constructing the structures of the aligned fragments; (4) modeling side chains and structures of the non-aligned parts, which are usually shaped as loops where no structural rules are imposed; (5) improving and evaluating the final structure of the target sequence.

Some well developed algorithms exist for the first step, template identification, that are served as benchmarks to any new approach. For example, BLAST (Altschul, 1990), which is a pair-wise sequence comparison method, records more than 30% accuracy. However, multiple sequence alignment, where the target sequence is aligned with more than one sequence, is far more accurate for detecting weak similarities between sequences (Notredame, 2002). Profile based approaches (Gribskov, 1987) and Hidden Markov Models (Krogh, 1996) excel in this field as well. PSI-BLAST (Altschul, 1997) introduced a position-specific scoring matrix. It scores position similarity between the target sequence and already existing sequences in PDB in an iterative manner until no new hits are found.

A recent tool that automates comparative modeling is TASSER-Lite tool (Pandit *et al.*, 2006). It relies on similar sequences and accordingly optimizes some parameters in order to decrease the computational time, while preserving prediction accuracy.

The disadvantages of this approach is that some proteins do not align – or align poorly – with other sequences of known structures in PDB. In addition, evolution might dramatically change sequences of proteins that were once very similar in both sequence and structure, and it is likely to have sequences with high degrees of similarities but with different structures.

1.1.2 Fold Recognition

Fold recognition assumes the existence of a similar template structure in protein databases for the target sequence. The reason for mapping to a template structure and not to the native structure is because there are more unknown sequences than known structures. Thus, this approach does not require the existence of similar sequences for the target sequence. The basic intuition in this approach is that proteins might have dramatically evolved by insertions, deletions, and substitutions in their amino acid sequences, but yet they maintained the same structure and the same function as they did before evolution. This intuition is seen in hemoglobin, which maintains the same structure in different species (Smith, 1995).

There are many techniques in fold recognition, however, there is no clear cut for these techniques as to which approach they belong. For example, one of the techniques in fold recognition is advanced sequence similarity methods. In this class there are methods like hidden Markov models (Krogh *et al.*, 1996), and PSI BLAST, which were recorded to be under the umbrella of comparative modeling approaches.

Another technique for fold recognition includes secondary structure prediction and comparison of the target sequence with sequences in databases. The accuracy of this model is based on the secondary structure prediction accuracy. An early approach in secondary structure prediction is the Chou-Fasman (Chou and Fasman, 1974) model. It uses statistical information from PDB. It calculates the frequency occurrence of an amino acid in a specific secondary structure in order to assign secondary structures for groups of amino acids using some rules. In fold recognition techniques, secondary structure information of the target sequence is combined with corresponding descriptors that might the scores for solvent accessibility of each amino acid. This approach predicts the 3D structure of the target sequence by predicting the descriptors of the target sequence and comparing them to the descriptors of the known structures (Przybylski and Rost, 2004).

Threading, another subclass in fold recognition, assumes the presence of a core structure in databases that threads with the target sequence. The threading of a sequence to a fold is evaluated by either environment-based or knowledge-based mean-force-potentials (Sikder and Zomaya, 2005) derived from PDB. It is shown that threading is a NP-hard problem (Garey and Johnson, 1979; Lathrop, 1994) in cases where every residue pairs are to be taken into account. Early approaches used simplified force-potentials calculated according to the surrounding environment of each amino acid and they used dynamic programming to evaluate the threading (Bowie *et al.*, 1991). Other approaches used the summation of pair-wise potentials for each and every atom in the sequence as the force-potential (Jones *et al.*, 1992) and applied double dynamic programming (Jones, 1998). A recent work (Skolnick *et al.*, 2004) developed an iterative advance. It first aligns the target sequence with core structures without calculating pair-wise potentials; in later iterations it uses the results of previous iterations to evaluate pair-wise interactions. Evaluation is done with a hybrid of three different functions, since many target sequences are capable of aligning with one structure. This helps in identifying accurate alignments and less accurate ones.

Fold recognition's efficiency, like comparative modeling, is restricted by the size of PDB. Since the mean-force-potential explores many features of a structure it is possible to find the optimal threading. However, there is no single approach until now that finds the optimal threading for more than half of the test cases. Nevertheless, this approach outperforms the comparative modeling for sequence similarities below 25% (Sikder and Zomaya, 2005).

1.1.3 *Ab initio* Approaches

Ab initio approaches do not rely on known structures in PDB. Instead, they generate a structure based on some thermodynamic principles between the atoms of the target protein. Many of these approaches simplify the structure of the protein in order to make the search space of structures manageable. Hydrophobic-polar (HP) and force field (FF) models fall under the *ab initio* category. This section discusses some methods in these two approaches.

1.1.3.1 Hydrophobic-Polar Methods

HP model simplifies the protein by assigning each amino acid to be a point in a 2D or 3D lattice. The intuition of this model is that hydrophobic amino acids are gathered in the core of a protein structure since they do not react with solvents. The output of this model is a valid 2D or 3D lattice (collision free) with maximum H-H contacts, which are buried in the core of a lattice. Each H-H contact is given the value -1; thus, the goal is to have a lattice with minimum energy, which is the summation of the H-H contacts. A genetic algorithm (GA) for the 2D HP model was developed (Unger and Moult, 1993). Individuals in a population are represented as sequences of letters defined over the alphabet {u, d, r, l} (up, down, right, and left), of length n – 1, where n is the size of the target protein. Genetic operators are applied on these strings to mutate and combine the strings. The operators try to maximize the H-H contacts, thus, minimize the energy. Another work (Bui and Sundarraj, 2005) for 2D HP model used secondary structures to assign directions. In this approach, the secondary structures of hydrophobic subsequences are chosen and a direction is assigned accordingly for each amino acid in the hydrophobic segments. For the other segments the approach randomly assigns directions.

The HP model is a simple model. While it is more efficient with peptides, proteins with small lengths, it does not scale well with proteins of long chain of amino acids, since the lattice becomes very complex, and hence hard to be kept valid.

1.1.3.2 Force Field Models

A FF model uses an energy objective function that evaluates the structure of a protein. This function attempts to represent the actual physical forces and chemical reactions occurring in a protein. Atoms are modeled as points in 3D with zero volume but with finite mass and charge, and bonds among atoms are modeled as Newtonian springs.

Schulze-Kremer (2000) introduced a GA approach for this model. In the energy model, an individual in a population is represented as a sequence of angles, with length 4n, where n is the number of amino acids in the target protein. GA in this work selects random angle values from PDB and applies controlled genetic operators, and tries to minimize the energy objective function. Cui *et al.* (1998) proposed another GA approach for the FF model. In their work, genetic operators select angles from the supersecondary structures angles library. Experiments in these two approaches lead to some interesting results. In the former approach, the root mean square deviation (RMSD) of the target protein against the reference protein reached to around 9 Angstroms (Å), which indicates a dubious relationship between the target structure and the reference structure. In the latter approach, the distance matrix error (DME) deviation of the target protein against the reference protein was between 1.48 and 4.48 Å relative to the sequence length.

GA's basic features proved to be more promising than other approaches applied in the literature of PSP problem for many reasons. A population in GA contains several structures, which is similar to running several Monte Carlo (MC) simulations, which start with one structure and manipulates it to reach the lowest energy. Since GA explores the search space all at once in a population, it avoids getting stuck with local optima that the MC simulations might get stuck with. The objective function in the FF model includes many physical forces and chemical reactions that mimic the molecular dynamics (MD) approach (Klepeis and Floudas, 2003). MD starts with a random structure, and it assumes that atoms move in a Newtonian manner under the influence of the different forces that they exert on each other. Because there are many moving atoms the error rate might be high in this approach; in addition, MD might also get stuck with local optima.

1.2 Thesis Objectives and Organization

In this thesis, we present a scatter search (SS) algorithm, an evolutionary search algorithm, for the PSP problem. To the best of our knowledge, there is no previous work for the PSP problem that used SS approach to structural search. Generated structures are evaluated by their energy potentials, which are used as factors to evaluate the computer simulated version of the PSP problem. The SS algorithm exhaustively searches the possible search space to find structures of proteins with low potential energies, which indicates a similarity between native structures and predicted structures. Our algorithm did produce structures with lower potential energy values than that of the real structures for the same energy function. For structural similarity between native structures and predicted structures, we use similarity measure, and our results are comparable to previous work (Schulze-Kremer, 2000) that used the same assumptions we used in our work.

The rest of the thesis is organized as follows. Chapter two is background knowledge of proteins and their compositions. In addition, it explains the energy potential function that is used to evaluate a protein structure. Chapter three gives an overview of the SS algorithm, and compares SS with GA. Chapter four describes our algorithm. Chapter five explains our experimental methodologies, presents our results, and discusses them. Chapter six concludes the thesis and presents some future works that will help improve our results.

Chapter 2 Problem Description

Approaches for PSP problem use low level information of the atoms and their surrounding solvents in order to generate native or native like 3D structures for proteins. Low level information deduce chemical reactions and physical forces among atoms. These natural reactions among atoms result in hierarchical folding process.

In this chapter, we first explain the atoms of amino acids and the bonds among the atoms. Next, we describe the levels of folding. We also illustrate the parameters involved in proteins. We explain the potential energy function, which evaluates the target structure. Finally, we explain how to represent atoms, which are points in space, as Cartesian coordinates.

2.1 Amino Acids and Peptide Bonds

An amino acid is constructed from one central atom, which is called the alpha carbon, $C\alpha$. To this $C\alpha$ atom an amino group (NH_2), a carboxy group (CO_2H), a hydrogen atom, and a side chain, R, are attached. A side chain is what differentiates amino acids from each other, and it can be as simple as one hydrogen atom as is the case of glycine amino acid or as complex as two carbon rings as is the case of tryptophan amino acid. When two amino acids are in contact with each other they form a peptide bond between the carbonyl C of amino acid i and amino N of amino acid i + 1. The result of the peptide bond is the H_2O molecule generated from the oxygen and hydrogen atoms of the carbonyl group and the hydrogen atom of the amino group (Setubal and Meidanis, 1997). Thus, all twenty amino acids have a common backbone shown in Figure 2.1.

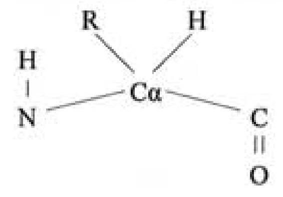

Figure 2.1: Backbone of an amino acid

2.2 Structural Hierarchy

Proteins undergo different levels of folding, and are categorized as four structures. The primary structure of a protein is formed from a chain of amino acids. The secondary structure of a protein is caused by hydrogen bonding of the backbone atoms, and the resulting structures are repeated sequences of helices, β-sheets, and loops. Carbonyl O of one amino acid is hydrogen bonded with H of amino N of another noncontiguous amino acid. Helices are the result of hydrogen bonding between atoms of two amino acids that are on average four amino acids apart. β-sheets are the result of hydrogen bonding between atoms of two amino acids of two backbone sequences, called β-strands, that are placed parallel or anti parallel with each other. Loops are the hooks that connect helices to helices, strands to strands, or helices to β-strands. They do not appear in the core of proteins as do the other two secondary structures. More complex structures of proteins are tertiary and quaternary structures. Tertiary or three-dimensional (3D) structures of proteins are formed as a result of different combinations of the secondary structures. Figure 2.2 shows the 3D structure of Crambin, a protein with 46 amino acids. Finally, quaternary structure of a protein is shaped by the combination of proteins with tertiary structures (Setubal and Meidanis, 1997).

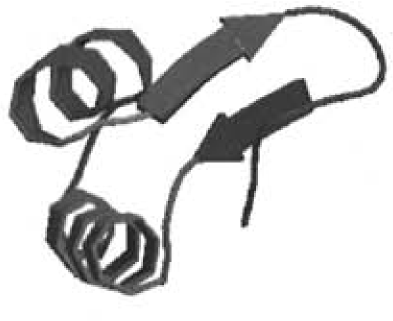
Figure 2.2: 3D structure of 1CRN. This figure is taken from Brookhaven database.

2.3 Parameters

Parameters involved in a protein are the result of the covalent bonds between atoms of the protein. The three parameters involved are bond length, bong angle, and torsion angle. The values of these parameters are based on the formation of the bonds among atoms and the nature of the bonds whether they are single, double, or partial double bonds. Most of the covalent bonds in the backbone of a protein are made by single bonds, where only a pair of electrons, each from an atom, is needed to give the electron octet shell to each of the two atoms. Double bonds appear in the side chains of a protein, where two pairs of electrons are needed to form the electron octet shell. Partial double bonds are the result of the peptide bonds between C of amino acid i and N of amino acid i+1. The result of a partial double bond is a double bond between carbonyl C and carbonyl O, and a single bond between C of amino acid i and N of amino acid i+1, and vice versa also appears in the repeating pattern of the backbone.

While bond lengths and bond angles are quite rigid, torsion angles, which are the angles between the normal vectors of two planes formed by three atoms, are flexible, thus, degrees of freedom are based on the torsion angles. There are four torsion angles present in a protein: phi φ, psi ψ, omega ω, and chi χ. Phi is the angle between the planes C-N-Cα and N-Cα-C, where N-Cα is the axis of rotation. This angle decides the distance of C-C of two amino acids. Psi is the angle between the planes N-Cα-C and Cα-C-N, where Cα-C is the axis of rotation. This angle decides the distance of N-N of two amino acids. Omega is the angle between the planes Cα-C-N and C-N-Cα, where C-N is the axis of rotation. This angle decides the distance of Cα-Cα of two amino acids. Finally, the chi angle is between the planes formed by the atoms of the side chains, and side chains can have as many as five chi angles depending on the length of the side chain. χ_1 is the angle formed by $N\text{-}C_\alpha\text{-}C_\beta\text{-}C_\gamma$ atoms, χ_2 is the angle formed by $C_\alpha\text{-}C_\beta\text{-}C_\gamma\text{-}C_\delta$ atoms, and so forth (Forman, 2001). As mentioned earlier, the torsion angles are the only degrees of freedom; therefore, they decide the structure of a protein. Secondary structures of a protein have specific valid ranges of phi, and psi angles, which are defined by the Ramachandran plot (Ramachandran and Sasiskharan, 1968). Regarding the omega angle, studies show that it either oscillates near 180° or 0° (Schulze-Kremer, 2000). The chi angle is a secondary rotation as a result of the phi and psi rotations. Therefore, proteins have 3D shapes because the bond between Cα and N atoms of one amino acid rotates the plane A in Figure 2.3 and the bond between Cα and C atoms of the same amino acid rotates the plane B in the figure.

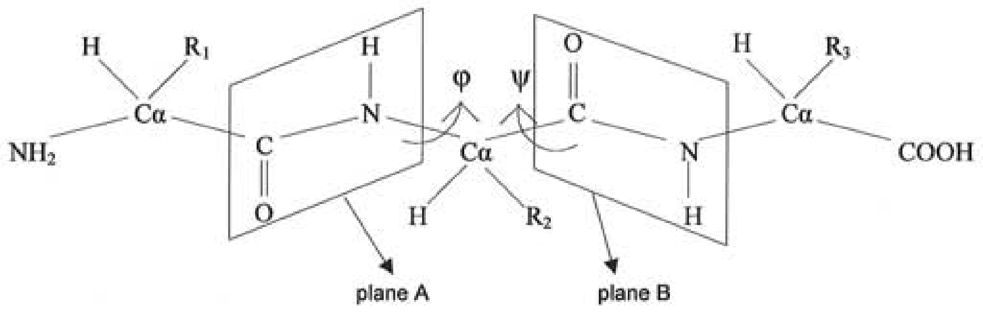

Figure 2.3: Segment of a protein backbone with planes of bonds Cα-N and Cα-C, plane A and plane B, respectively.

2.4 Force Field

The objective function used in our implementation is a simplified version of the force field introduced in the program Chemistry at HARvard Molecular Mechanics (CHARMM) (Brooks *et al.*, 1983). The force field used in the CHARMM model calculates the potential energy of a protein structure. The challenge in calculating the energy objective function is to minimize this energy, which is a path in determining the structure of a protein. In addition, it is used as a factor to evaluate the computer simulated version of the PSP problem.

The potential energy function of the original CHARMM model is divided into internal or bonded energies, and external or nonbonded energies. Equation (1) is the potential energy function, E, as a function of the conformation, c:

$$E(\bar{c}) = \sum_{bonds} K_b(b-b_o)^2 + \sum_{angles} K_\theta(\theta-\theta_o)^2 + \sum_{improper\ torsions} K_{imp}(\varphi-\varphi_o)^2 + \sum_{torsions} K_\chi(1+\cos(n\chi-\delta)) + \sum_{hydrogen}\left(\frac{A_{ij}}{r_{ij}^{10}} - \frac{B_{ij}}{r_{ij}^{12}}\right) + \sum_{electrostatic}\frac{q_i q_j}{4\pi\varepsilon_o\varepsilon_r r_{ij}} + \sum_{van\ der\ Waals} 4\varepsilon_{ij}\left(\frac{\sigma_{ij}{}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}{}^{6}}{r_{ij}^{6}}\right) \qquad (1)$$

Internal energies of the potential energy function include bond, angle, improper torsion, and torsion energy. External energies include the electrostatic and van der Waals components which are calculated between atoms separated by two (1,3-pairs) or three (1,4-pairs) covalent bonds. Hydrogen bond energy is between hydrogen acceptors (mostly oxygen) and hydrogen donors (nitrogen), which are on average four amino acids apart.

The first term of Equation (1) is the summation of bond stretches of the target protein from the ideal bond lengths. b_o is the ideal bond length between specific bonded atoms, while b is the actual bond length in the predicted conformation. K_b is the bond force constant that determines the strength of the bond. The second term of Equation (1) is the summation of bond bending of the target protein from the actual bong angles. θ_o is the ideal bong angle between three bonded atoms, while θ is the actual measured angle in the predicted conformation. K_θ is the angle force constant that determines the elasticity of the bending angle. The third term of Equation (1) is the summation of out of plane bending. φ_o is the ideal torsion angle, φ is the torsion angle of the predicted conformation, and K_{imp} is the improper torsion force constant. These three components of the potential energy function correspond to the difference in geometry of the predicted conformation from the real conformation of the target protein. Thus, in cases where the predicted conformation is optimum, the values of these three terms are close to zero. The fourth term of Equation (1) is the torsion angle function. This function represents a rotation around the middle bond of 1,4-pair atoms depicted by a torsion angle χ and multiplicity $n=1,2,3,4,6$. The purpose of this method is to prevent steric clashes between 1,4-pair atoms, and thus, it can be modeled as a barrier. The torsion energy function is modeled as a periodic function. $K\chi$ is the torsion force constant, and δ is the phase angle. The values of the torsion force constants for each set of four connected atoms are tabulated (Jorgensen *et al.*, 1996).

For the non bonded atoms, the electrostatic energy is modeled by Coulomb potential. It calculates the force exerted from the interaction of the charges of two non bonded atoms. The role of this function is important in forming the structure of a protein since it models the interaction of two charged atoms. q_i and q_j are the charges of the two atoms. ε_o is the permittivity of free space, ε_r is the dielectric constant, and r_{ij} is the Euclidean distance between these two atoms. The van der Waals term of Equation (1) uses Lennard-Jones 6-12 potential. This force depicts the attractive and repulsive forces between atoms. The repulsive force is the result of electron collision; this force is generated when two atoms are in close proximity to each other. The repulsive force is represented in the van der Waals function as $\sigma_{ij}^{12} / r_{ij}^{12}$, where σ_{ij} is the van der Waals radii, and r_{ij} is the Euclidean distance between atoms *i* and *j*. The attractive force is the result of variation of charges in the electron clouds of one atom, where this variation leads to the generation of a dipole in this atom and in another atom, thus inducing an attractive force among these two atoms. This force arises when two atoms are in long range. The attractive force is represented in the van der Waals function as $\sigma_{ij}^{6} / r_{ij}^{6}$. For long distances between atoms the attractive force is dominant, whereas for short distances the repulsive force is dominant. This phenomenon minimizes the total potential energy function if atoms are placed in optimal distances. For example, if atoms are placed in a linear fashion the attractive force is dominant; thus, the van der Waals term is very small and negative. This value is small compared to the large value of the electrostatic function. Therefore, it will not help in decreasing the total potential energy. In van der Waals function, ε_{ij} is the van der Waals depth. The values of both van der Waals sigma and depth are also tabulated (Jorgensen *et al.*, 1996).

2.5 Torsion to Cartesian

A protein is composed of a chain of 20 different amino acids. Various representations exist for such chains. We are interested in two representations: the torsion representation, where amino acids are characterized by the torsion angles described earlier in this chapter, and the Cartesian representation, where each atom in a protein is characterized by its Cartesian coordinates that identifies its location. The parameters of these representations, the torsion and the Cartesian coordinates, are primarily used to: (1) identify protein's properties, such as RMSD and potential energy; (2) enable protein structural analysis.

These two representations are: (1) similar, i.e., each can be described as the function of the other; (2) exchangeable, i.e., can be interchanged when performing a computation on the protein based on the complexity of the computation. For example, the torsion angle representation provides a relatively small finite domain for randomly building proteins, whereas the Cartesian coordinates provide sufficient information about each atom's location, which enables efficient computation of energies and RMSDs.

The conventional way to transform between the two representations follows directly from multidimensional mathematics (Parsons *et al.*, 2005). To illustrate, consider the set of bonded atoms in Figure 2.4. Given the coordinates of the three atoms A, B, and C, the bond length of the bond C-D, the bond angle between the points B-C-D, and the torsion angle in accordance to the bond B-C, all are used to compute the coordinates of the point D as follows: (1) by placing point D_0 of bond length C-D away from the point C in the same axis as the bond B-C; (2) rotating D_0 with an angle equivalent to the bond angle B-C-D to find D_1; (3) rotating D_1 along the bond B-C with an angle equivalent to the given torsion angle to determine the position of D2.

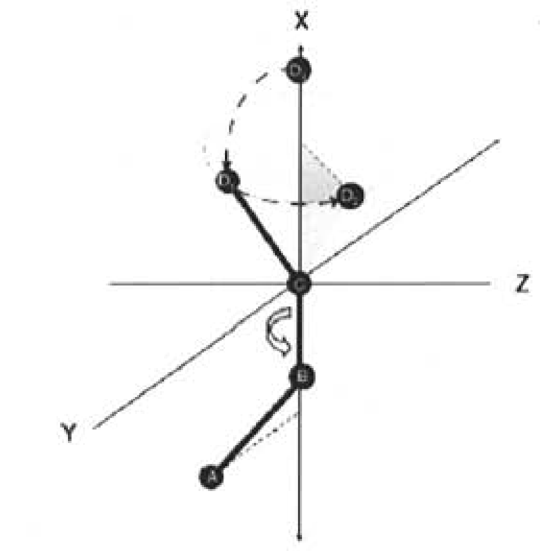

Figure 2.4: Torsion to Cartesian transformation. This figure is taken from Parsons *et al.*, (2005).

The above example describes how to compute the coordinates of an atom given the coordinates of the three previous atoms, the bond lengths, the bond angles, and the torsion angles. To transform from the torsion representation to the Cartesian representation, the same computation is repeated along the amino acid chain. To illustrate, given the coordinates of the first three atoms in the chain, as well as, the relative bond length, bond angle, and torsion angle, the coordinates of the forth atom are computed. Then given the coordinates of the last three atoms, the coordinates of the fifth atom can be then computed and so on.

Chapter 3 Background on Scatter Search

Scatter search (SS) is an interesting evolutionary search strategy. Its initial formulation dates back to the 1970s as a result of a combination of decision rules and problem constraints. SS proved to achieve interesting results for different optimization problems (Laguna and Marti, 2003). It uses search strategies to produce and maintain diverse solutions, and to explore the search space extensively through controlled randomization. In this work, we used SS implementation for PSP problem.

In this chapter, we first describe the basic SS algorithm. Next, we describe how we represented the solutions in our implementation. We compare SS and GA. We mention the assumptions that we made in our implementation. Finally, we illustrate our implementation of SS for the PSP problem.

3.1 Scatter Search Design

Glover (1998) introduced the first outline of SS algorithm. Prior to this outline, SS was incorporated into tabu search algorithm. The sketch of the general SS outline is explained in the following five steps (Laguna and Marti, 2003):

1. Step 1 generates an initial set of solutions by employing strategies to guarantee diversity and randomness. Frequency-based memory is one strategy that produces solutions that span the whole search space, and chooses parameters from the pool of possible parameters for the solutions in the population in an equally balanced manner. In this step, or in any part of the SS algorithm, solutions that are newly generated do not have to be feasible. This allows the algorithm to explore more choices, which exhausts the search space. This step is referred to as Diversification Generation Method.

2. Step 2 in the outline of SS is the Improvement Method. This step improves every solution given as input to the Improvement Method, and its implementation is problem-specific. The resulting solution is not necessary to be a feasible one, although most probably it will be. In case the improved solution is not better than the input solution in terms of the objective function, which again is problem-specific, the input solution is considered in the following steps.

3. SS operates on a fairly small set of solutions. This set is called reference set. Reference set contains solutions that are arguably the best solutions. Best in this step is not only restricted to solutions that have best objective function values, but also to solutions that introduce diversity to the reference set. In the initial iteration of the SS, this step chooses b_1 best solutions from the initial population generated from step 1. The remaining b_2 solutions that enter the reference set are the ones that introduce diversity to the reference set. In this case, b_2 solutions are chosen from the initial population, such that, they are diverse from the best b_1 chosen solutions. Diversity is implemented by choosing solutions that have rather far objective function values from the best chosen solutions. The resulting reference set is of size $b = b_1 + b_2$, which, in most cases, is 20% of the initial population size. In later iterations the reference set is updated to again include "best" solutions; hence, this step is called Reference Set Update Method.

4. The solutions in the reference set are manipulated by SS operators. In addition to the Improvement Method, SS performs a combination operator. The combination operator combines two or more solutions together to yield another new solution. Prior to the combination step a method exists to generate subsets of solutions that are to be combined. This step decides which solutions to combine; hence, it is called Subset Generation Method. A subset in this sense is the subset of solutions to combine, which might be of size two, three, four, or from five up to b, which is the size of the reference set. These subsets of different sizes are given the names subset type 1, subset type 2, subset type 3, and subset type 4, respectively. Subset type 1 generates subsets of every two solutions. Care is

taken here and in all subset types not to include the same subset in later iterations; i.e., in later iterations only new solutions gain admission to subsets. Subset type 2 augments a third solution to subset type 1. Subset type 3 augments a fourth solution to subset type 2. Finally, subset type 4 augments up to b – 4 solutions to subset type 3, which makes the whole reference set.

These subsets guarantee combinations of solutions "within clusters" and "across clusters". For example, if subset type 1 combines two high quality solutions in terms of the objective function, implementing subset type 3 augments to these two high quality solutions a diverse solution. This is one of the key features in SS, which proves the capability of SS to exhaust the search space with every possible combination.

5. A Solution Combination Method combines the solutions in the subsets created by the Subset Generation Method to yield one or more new solutions for each combination. Its implementation is problem-specific. However, a common issue among all solution combination methods is to extrapolate parameters that are not considered in the current solutions or in the pool of possible parameters for the solutions. Following are three types of solution combination methods that are used in three different applications (Laguna and Marti, 2003):

    - Linear combination: a SS implementation for an instance of unconstrained nonlinear optimization problem used linear combination for the Solution Combination Method. The problem can be put into the form

      $$\begin{array}{ll} \text{Minimize} & f(x), \\ \text{subject to} & l \le x \le u. \end{array}$$

      The number of resulting combined solutions is based on the quality of the reference solutions. In this case three trial solutions are created from two reference solutions. Following are the three trial solutions, where x' and x'' are the reference solutions:

$$\text{Combination } 1: x = x' - d$$
$$\text{Combination } 2: x = x' + d$$
$$\text{Combination } 3: x = x'' + d\ ,$$

where $d = r\frac{x'' - x'}{2}$ and r is a random double between 0 and 1.

- Score-based combination: a SS implementation for an instance of knapsack problems, 0-1 knapsack, used score-based combination for the Solution Combination Method. The problem is mathematically expressed as follows:

$$\begin{aligned} &\text{Maximize} && \sum_i c_i x_i\ , \\ &\text{subject to} && \sum_i a_i x_i \le c, \\ & && \text{with } x_i \in \{0,1\} \quad \forall i, \end{aligned}$$

where c_i is the ith profit coefficient, x_i is the ith variable that is either 0 or 1, a_i is the ith weight coefficient, and c is the total weight.

The Solution Combination Method in the context of 0-1 knapsack problem is probabilistic. Each variable in the resulting trial solution has a score, and according to this score a decision is made whether variable x_i should be 1 or 0. The score for variable x_i is calculated according to the objective function values of the two reference solutions j and k, which is calculated as follows:

$$score(i) = \frac{ObjVal(j)x_i^j + ObjVal(k)x_i^k}{ObjVal(j) + ObjVal(k)}$$

where *ObjVal(j)* is the objective function value of solution j, and x_i^j is either 1 or 0, which is the value of variable x_i in solution j. The decision of the ith variable in the resulting solution is implemented as follows:

$$x_i = \begin{cases} 0 & \text{if } r \le score(i) \\ 1 & \text{if } r > score(i) \end{cases}$$

where r is a random double between 0 and 1.

- Combinations by votes: a SS implementation of the linear ordering problem (LOP) used combination by vote for the implementation of the Solution Combination Method. A solution in the LOP is a permutation p of size m consisting of indices of a $m \times m$ matrix of weights $E = \{e_{ij}\}_{m \times m}$. LOP tries to find p that maximizes the sum of the all weights in the upper triangle of the matrix, triangle above the main diagonal. Mathematically the problem tries to maximize the following equation:

  $$C_E(p) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} e_{p(i)p(j)}$$

  where $p(i)$ and $p(i)$ are the row and column indices of the matrix respectively.

  Voting combination methods try to reduce diversity between the combined solution and its reference solutions. Each solution in the subset votes for its first element that did not yet appear in the resulting solution. If the entire reference solutions in a subset vote for the same element, the Solution Combination Method chooses this element for the next available position in the resulting combined solution. If the reference solutions vote for different elements, then there exist two strategies to select from the voted elements for the next available position in the resulting combined solution. The first strategy always selects the one with higher partial objective function value. The second strategy first selects the one with higher partial objective function value and in later positions it selects the one with lower position value.

In addition to these solution combination methods versions of crossover implemented in GAs can also be used in solution combination methods. However, the argument in this case is that crossover might lose track of some good elements in solutions, thus, it might lose high quality solutions.

Figure 3.1 depicts the implementation of SS; it is taken from Laguna and Marti, (2003).

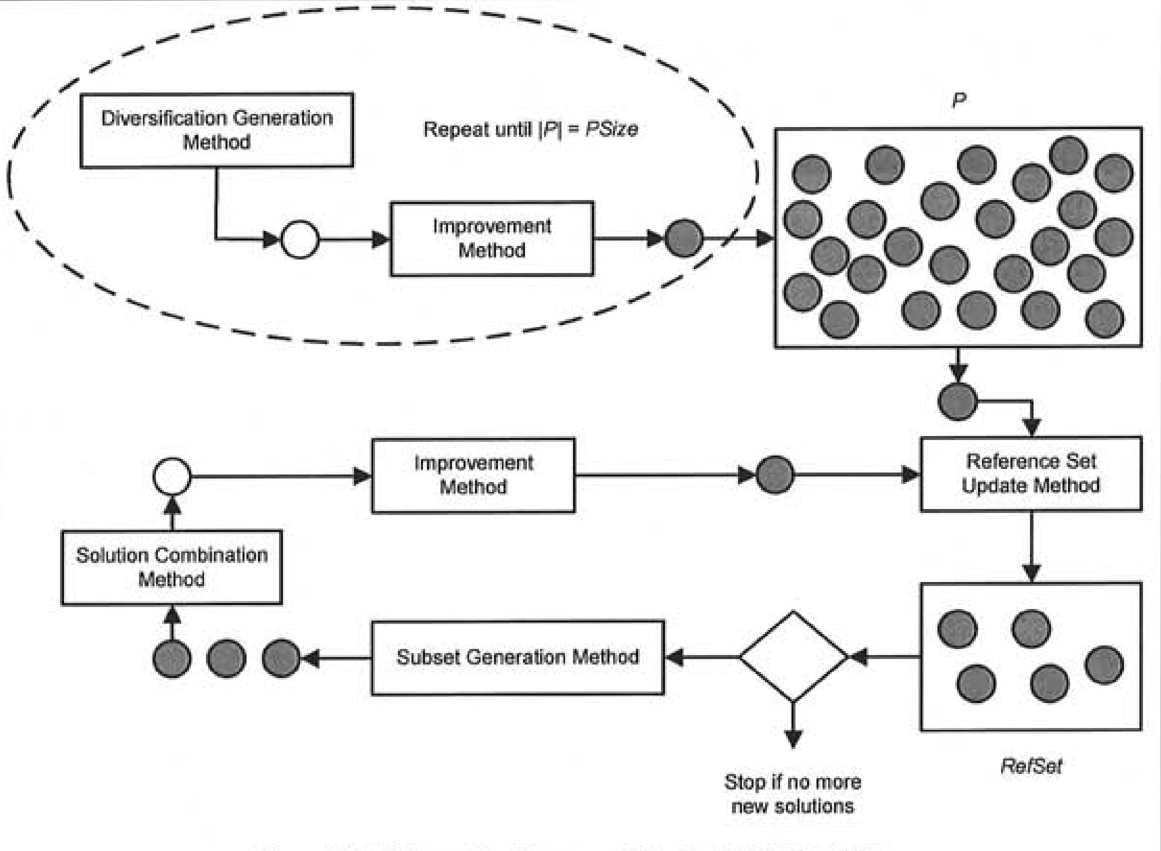
Diversification Generation Method
Repeat until |P| = PSize
Improvement Method
P
Improvement Method
Reference Set Update Method
Solution Combination Method
Subset Generation Method
Stop if no more new solutions
RefSet

Figure 3.1: Schematic diagram of the basic SS algorithm

The SS algorithm starts by generating the initial population, *P*, by applying the Diversification Generation Method, and each solution in *P* is subjected to the Improvement Method. Solutions in Figure 3.1 are depicted as circles. Improved solutions are depicted as dark circles. The Reference Set Update Method initially generates the initial reference set, *RefSet*, from *P*. As mentioned above, *RefSet* size is usually 20% the size of *P*, *PSize*, and stays as such throughout all iterations. Subsets of the reference solutions are built by the Subset Generation Method, and the reference solutions in each subset are combined by the Solution Combination Method. Each resulting solution from the Combination Method is again subjected to the Improvement Method. The improved solutions are added to *RefSet* in case they are "better" than the solutions in *RefSet*. This step is implemented in the Reference Set Update Method. The notion "better" here is not only restricted to high quality solutions but to more diverse solutions as well. The algorithm is terminated when there are no more new solutions to be added to *RefSet*.

3.2 Scatter Search and Genetic Algorithm Comparison

GA is another population-based evolutionary search strategy. Throughout the years many instances of GA were introduced. The basic GA's design has some differences with the basic SS's design. Following are the differences between the two search strategies:

1. SS operates on reference set with size 20% of the GA's population size.
2. SS applies combination and improvement methods on predetermined subsets of reference solutions, while GA probabilistically selects parents from the population to apply crossover and mutation.
3. Reference set in SS is updated by deterministic rules that are implemented in Reference Set Update Method, while a population in GA is updated by probabilistic rules that apply "survival of the fittest" attitude.
4. SS uses local search in combination and improvement methods to further investigate solutions' neighborhoods, while this feature was added later to GAs to generate hybrid forms of GA operators to better improve the quality of the solutions.
5. Solution combination methods in SS combine two or more solutions, while crossover operators in GA combine only two solutions.
6. In generating the initial population, SS uses controlled randomization to introduce diversity in the population, while GAs use full randomization in generating the initial population.

Chapter 4 Scatter Search Algorithm for Protein Structure Prediction Problem

Our proposed solution is an adaptation of the SS approach for the PSP problem. In this chapter, we describe the steps of the SS algorithm relative to this problem. In our discussion, we use an example of a small protein, Crambin with PDB ID 1CRN.

4.1 Assumptions

Our solution of the PSP problem assumes constant binding geometry, where all bond lengths and bond angles are constant, i.e., bond lengths and bond angles are not affected by SS operators. This cancels bond, angle, and improper torsion components of the potential energy function of Equation (1). This is not the case with real proteins where the values of the bond lengths and bond angles are related to the changes of the torsion angles; however, degrees of freedom introduced by the torsion angles are enough to result in real structures of proteins with some small root mean square deviations. In addition, hydrogen atoms of both amino nitrogen and alpha carbon are also discarded, thus, the hydrogen energy component in Equation (1) is also canceled. The resulting potential energy function is simplified to the Equation (2).

$$E(\bar{c}) = \sum_{torsions} K_\chi (1 + \cos(n\chi - \delta)) + \sum_{electrostatic} \frac{q_i q_j}{4\pi\varepsilon_o \varepsilon_r r_{ij}} + \sum_{van\ der\ waals} 4\varepsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}} \right) \qquad (2)$$

Every amino acid has 2 non-bonded atoms, and each two contiguous amino acids have 6 non-bonded atoms. Since all the components of the above energy function are calculated between atoms that are three covalent bonds apart, there are 8m – 6 computations for m amino acids. Therefore, the complexity of the energy function is $O(m)$.

Side chains in our solutions are represented as a single carbon atom, C_β. This assumption does not represent a protein as a whole; however, it represents the backbone of a protein. Therefore, each amino acid in our solution has only five atoms: N, $C\alpha$, C_β, C, and O. These assumptions have normally been used in previous works (Schulze-Kremer, 2000).

4.2 Solution Representation

A protein is represented by a list of successive objects. An object represents an amino acid in the backbone of the peptide chain, and its position in the list is the same as that of the amino acid in the chain. Hence, the size of the list is equal to the number of amino acids in the chain. Each object stores the type and characteristics of the atoms of the corresponding amino acid. The characteristics of the atoms are: the partial charge, van der Waals epsilon, and van der Waals sigma of the corresponding atom. Each object in the list also contains values for the torsion angles phi, psi, and omega.

A target solution for the PSP problem is a 3D protein structure. The protein structure produced by our SS algorithm is henceforth referred to as the target protein. It is given by the values of the torsion angles in the list of objects, from which Cartesian coordinates can be derived. For example, a peptide with three amino acids (AA) has three object fields in its list: {AA1, AA2, AA3}. AA1 object has nine attributes in this case: {THR, 0, 147.7, 178.9, ATOM1, ATOM2, ATOM3, ATOM4, ATOM5}. THR is shorthand for Threonine, which is the amino acid's name; the three values are the phi, psi, and omega values of this amino acid, respectively. ATOM1 object has five attributes: {N, -0.76, 0.17, 3.25, POSITION}. N is shorthand for nitrogen; the three values are the partial charge, van der Waals epsilon, and van der Waals sigma of nitrogen, respectively. POSITION object has three attributes: {17.047, 14.099, 3.625}, which are the Cartesian coordinates of the nitrogen atom derived from the torsion angles.

4.3 Diversification Generation Method

Our Diversification Generation Method produces random and diverse initial solutions. Because a solution in our approach is a protein, it is characterized by its potential energy value. For physical systems, the lower the energy, the more stable the system. Therefore, a high quality solution is a solution that has a low potential energy value. The Diversification Generation Method has no restriction to generate high quality solutions; therefore, the solutions produced in the initial population are random solutions. On the other hand, once an initial set of solutions are generated, the method generates another set of solutions that are diverse from the initial set by their potential energy values. Moreover, the method always generates feasible solutions in terms of the torsion angles, because the repositories from which torsion angle values are selected from are formed from PDB. The total population size of candidate solutions is assumed to be $|P| = 100$.

Controlled randomization is employed in selecting phi, psi, and omega values for each amino acid from the ten most occurring 10 degrees intervals of 100 proteins with known structures. These values are collected manually from PDB. A 10 degrees interval is not restricted to be an interval that starts with a value that is a multiple of 10 or ends with a value that is a multiple of 10. It can be any 10 degrees interval that has the highest number of occurrences in PDB. Each of the torsion angles has ten bins. A bin contains angle values, and the exact number of angle values in a bin depends on PDB data. A 10 degrees interval is enough to provide us with a wide range of angle values, in addition, its sets a limit on the Diversification Method not to include values that occur occasionally. The latter point is also satisfied by selecting the ten most occurring intervals. Since a protein's 3D structure in our approach is mainly controlled by the torsion angles, in a search tool like SS it is important to provide the SS operators with wide range of values to exhaust the search. For this reason we are creating ten bins for each torsion angle. Table 4.1 shows one of the ten bins of the phi torsion angle, the range of values for this bin is [-121, -131]. The bin contains 54 values, and as mentioned earlier the number of values depends on PDB data.

Table 4.1: A phi bin of range [-121, -131]

Phi angle values					
-128.7	-124.5	-123.3	-125.4	-125.3	-123.9
-126.3	-125.7	-129.7	-127.6	-130.0	-126.0
-124.9	-126.6	-124.2	-127.7	-129.8	-124.8
-126.8	-128.9	-129.3	-130.5	-128.4	-125.1
-130.8	-122.4	-124.1	-122.8	-126.4	-121.0
-129.5	-122.3	-122.6	-122.1	-123.5	-127.3
-122.9	-130.2	-121.6	-121.4	-121.2	-126.9
-128.1	-123.8	-121.8	-130.3	-129.2	-127.4
-127.1	-130.7	-121.9	-125.6	-127.9	-123.1

To achieve diversity, our Diversification Generation Method uses frequency-based memory. The frequency-based memory in the context of PSP problem keeps track of the number of times a torsion angle value is chosen from a specific bin for a specific amino acid in all the solutions in *P*. The goal of this strategy is to divide the counts in a fair manner among the bins for a specific amino acid. For example, Table 4.2 shows how many times a specific bin is chosen for the first 10 amino acids of 1CRN. Column one shows the first ten amino acids (AA) of 1CRN, row one shows the ten selected bins.

Table 4.2: Frequency counts of the selected bins for the first 10 amino acids of 1CRN

	Bin1	Bin2	Bin3	Bin4	Bin5	Bin6	Bin7	Bin8	Bin9	Bin10
AA1	0	0	0	0	0	0	0	0	0	0
AA2	11	4	7	12	12	8	17	6	16	7
AA3	8	11	9	12	12	9	13	6	6	14
AA4	5	15	13	9	7	7	11	8	15	10
AA5	12	11	11	9	13	12	9	10	9	4
AA6	9	10	10	12	9	16	8	8	8	10
AA7	11	6	12	11	11	7	12	9	12	9
AA8	9	17	10	14	11	6	8	11	9	5
AA9	7	13	9	7	13	15	11	8	11	6
AA10	6	7	10	4	13	12	17	10	11	10

Because we have 100 solutions in the initial population, the total frequency counts of the selected bins for an amino acid is 100. The phi, psi, and omega values of the first amino acids of every candidate solution in the population are the phi, psi, and omega values of the reference protein, which is the native protein under study. Its torsion angle values are the values that are recorded in PDB. For this reason, the Diversification Method did not choose bins for the first amino acids; this is shown in Table 4.2 where all the frequency counts of the AA1 are zeros. The key idea in keeping the real torsion angle values for the first amino acids is because of the torsion to Cartesian conversion process. We are keeping the Cartesian coordinates of the first five atoms, which are the atoms of the first amino acid, the same as the reference protein. The key insight is to allocate the rest of the atoms of the target protein according to the first five atoms so that the reference points of both the reference protein and the target protein are compatible.

The process of assignment values to phi, psi, and omega has two parts: the first part is to select a bin, and the second part is to select a value from this bin.

To implement the first part of the assigning process, we use three $m \times n$ freqCount matrices, where m is the number of amino acids of the target protein, and n is the number of bins, which in our approach is ten. Each of the three matrices refers to a torsion angle, and the number of times a bin is selected for an amino acid is stored in the freqCount matrix. The probability of selecting a bin for an amino acid is inversely proportional to the frequency count of this bin with respect to the amino acid.

Figure 4.1 presents an algorithm for assigning a torsion angle value for an amino acid, aminoAcidIndex. Initially every element of a freqCount matrix is initialized to zero. The algorithm first computes the total frequency counts of the ten selected bins for an aminoAcidIndex in lines 2-3. This sum is then subtracted from each amino acid-bin element to be stored in the rFreq array in line 6, which holds the frequency complements of an amino acid with respect to each bin. The variable total in line 7 holds the sum of the frequency complements in the rFreq array. In case the variable total equals to zero, which is the case of the first call of this method, the algorithm randomly selects a bin in line 10 and increments the frequency count of this bin in line 15. After the first call, the choice of selecting a bin is based on the total variable and the rFreq array, which is shown in lines 12-14. The second part of the assignment process is completely random, i.e., once a bin is selected, the choice of a torsion angle value for an amino acid from this bin is random. The algorithm returns any random value from this bin in line 16.

Each call to GET-TORSION-VALUE costs $O(n)$ time, and there are $O(3m)$ such calls for a solution of m amino acids with three torsion angles each. The total cost of GET-TORSION-VALUE is therefore $O(nm)$ for a solution.

```
GET-TORSION-VALUE(aminoAcidIndex)
1.      sum ← 0
2.      for i ← 0 to n
3.              sum ← sum + freqCount[aminoAcidIndex][i]

4.      total ← 0
5.      for j ← 0 to n
6.              rFreq[j] ← sum - freqCount[aminoAcidIndex][j]
7.              total ← total + rFreq[j]

8.      index ← 0
9.      if total = 0
10.             index ← randomInteger(0, n – 1)
11.     else
12.             k ← randomInteger(0, total – 1)
13.             while k > rFreq[index]
14.                     k ← k - rFreq[index++]

15.     freqCount[aminoAcidIndex][index]++
16.     return value from the interval of index index
```

Figure 4.1: Pseudocode for assigning a torsion angle value for an amino acid

After generating a solution, the Diversification Generation Method checks whether the generated solution is feasible in terms of the geometric placements of the atoms. For non-bonded atoms, the minimum distance between them should be 1Å. For bonded atoms, the minimum distance should be 0.5Å. In case these conditions are not satisfied for the solution in hand, the method discards the solution and generates another solution. The method generates $|P|$ valid solutions.

Every amino acid has 7 bonded and 2 non-bonded atoms. Therefore, if we take amino acid by amino acid the running time of checking whether a solution with *m* amino acids is a valid one is $O(9m)$. For contiguous amino acids there are 4 bonded and 6 non-bonded atoms. Therefore, for all amino acids that are contiguous the running time is $O(10(m-1))$. The total cost of the validation process is therefore $O(m)$.

Table 4.3 shows ten solution candidates from *P* with their energy values. These solutions are selected after *P* is sorted in an increasing order based on the potential energy function values of the solutions. The first five solutions are the first five solutions in *P*, and the next five solutions are the last five solutions in *P*.

Table 4.3: Energy values of the first five and the last five solutions in the initial population

Solution	Energy	Solution	Energy
1	3.95E+07	6	6.18E+13
2	4.29E+07	7	7.72E+14
3	5.28E+07	8	8.15E+14
4	5.98E+07	9	2.19E+15
5	8.28E+07	10	5.23E+15

4.4 Improvement Method

Since the phi, psi, and omega values of the solutions generated by the Diversification Generation Method are taken from 100 proteins collected from PDB, the Improvement Method always begins with feasible solutions regarding the values of the torsion angles.

Our Improvement Method improves every solution in a set of solutions subjected to the Improvement Method. Therefore, every solution in the initial population and *RefSets* are improved by the Improvement Method. The improvement technique used in our Improvement Method is applied on the torsion angles of the target protein; this is because the torsion angles determine the Cartesian coordinates of the atoms of a protein. Thus, the structure of a protein is changed by changing the torsion angles.

At the beginning of the SS iterations, only 40 percent of the torsion angles of a solution are subjected to improvement. These angles are selected randomly. However, in subsequent iterations the percentage of the torsion angles chosen for improvement decreases; this is because a solution converges in later iterations, so we do not want to alter any good solution in the last iterations.

The improvement applied on a torsion angle is rather a simple one. For example, if the angle to be improved is a phi torsion angle, the method randomly selects a feasible value from the ten bins, mentioned in Section 4.3, corresponding to the phi angle.

The improved solution replaces an existing solution in a set of solutions if its potential energy value is lower. Then the method checks whether all the solutions in the set are valid solutions in terms of the geometric placements of the atoms. For non-bonded atoms, the minimum distance between them should be 1Å. For bonded atoms, the minimum distance should be 0.5Å. In case these conditions are not satisfied for the solution in hand, the Improvement Method tries to improve the invalid solution for five times until a valid solution is produced. If after five attempts the method does not generate a valid solution, the improved solution is discarded and the existing one is kept.

Only 40% of the torsion angles of a solution are improved. Therefore, there are $O(6m/5)$ replacements for a solution with m amino acids. Moreover, the running time of the solution validation check is $O(m)$, therefore, the total cost of the Improvement Method for a solution is $O(m)$.

4.5 Reference Set Update Method

Our *RefSet* contains two sets of solutions. The first set contains high quality solutions, HQRefSet, and the second set contains diverse solutions, divRefSet. The size of the *RefSet* is $b = b_1 + b_2$, where b_1 is the size of HQRefSet, and b_2 is the size of divRefSet. We select b to be 20 percent of P's size. Therefore, *RefSet* contains 20 solutions, and we select $b_1 = b_2 = 10$. Solutions in the initial *RefSet* are taken from P, which is generated by the Diversification Generation Method, and improved by the Improvement Method. Initially, b_1 solutions with minimum energy values are chosen from *P*. The b_2 diverse solutions are solutions that have diverse energy values from the b_1 high quality solutions.

In order to choose b_2 diverse solutions, we first need to define a diversity measure. In our problem a solution is diverse from another solution if their structures are diverse. Because the potential energy of a solution reflects its structure, the diversity measure is based on the potential energy, thus, the torsion angles of a solution are the parameters of the diversity measure. After selecting b_1 best solutions and including them in *RefSet*, for each unselected solution in *P* and all the solutions already in *RefSet*, we calculate the number of torsion angles that are different within $\pm 5^\circ$. For each unselected solution in *P*, we save the solution in *RefSet* that has the minimum number of differences. Then, from all unselected solutions in *P*, we select the solution that has the maximum of these minima to be a member in *RefSet*. Once a new solution is added to *RefSet*, the algorithm updates the minima set and repeats the same procedure until |*RefSet*| equals *b*. The key insight in saving the minimum differences is to control diversity. Moreover, selecting the solution that has the maximum of these minima assures enough diversity in the *RefSet*.

Table 4.4 shows the initial *RefSet* of the 1CRN example. Columns one and three show the solution number corresponding to the *RefSet*, while columns two and four show their potential energy values, respectively. The first ten solutions are the solutions in the HQRefSet, and the second ten solutions are the solutions in the divRefSet.

Table 4.4: Solutions in the initial *RefSet* of the 1CRN example

Solution	Energy	Solution	Energy
1	3.95E+07	11	3.47E+10
2	4.29E+07	12	8.24E+10
3	5.28E+07	13	9.12E+10
4	5.98E+07	14	2.84E+11
5	8.28E+07	15	7.38E+11
6	8.80E+07	16	3.82E+12
7	9.18E+08	17	4.82E+13
8	9.35E+08	18	7.34E+13
9	9.75E+08	19	2.84E+15
10	2.38E+09	20	3.45E+15

The Reference Set Update Method is applied to the initial population *P* and to each of the output solutions, output_solution, of the Solution Combination Method (after improvement). For the latter case, there are two criteria for a solution to gain membership to *RefSet*. Figure 4.2 presents an algorithm that takes an output_solution as an input and checks whether this solution can replace an existing solution in *RefSet*. The algorithm replaces a solution in HQRefSet, with the output_solution if the energy of the output_solution, output_energy, is less than the energy, HQ_worst_energy, of the worst solution in the HQRefSet in lines 1-2. If this condition is not satisfied, the algorithm checks if the output_solution is more diverse than the least diverse solution, least_diverse_sol, in divRefSet.

To check whether the output_solution is more diverse than the least_diverse_sol, we first discuss the algorithm of Figure 4.3. The algorithm takes as input a solution, which is a list of amino acid objects, and a solution pool, which is a list of solution objects. For each solution in the solution pool, the algorithm counts the number of torsion angles that differ within $\pm 5^\circ$ with the input solution. In lines 13-15, the algorithm saves the count that is the minimum corresponding to the input solution pool, and it returns this minimum. REFSET-UPDATE-METHOD calls the MINIMUM-DIFFERENCE twice. The call in line 4 sends as arguments an output_solution and HQRefSet, thus, it returns the minimum count of torsion angles that differ within $\pm 5^\circ$ of the output_solution and a solution in the HQRefSet. The call in line 5 does the same for the least_diverse_sol and the HQRefSet. In lines 6-7, the algorithm switches the output-solution with the least_diverse_sol if the minimum count of the output-solution and the solution in the HQRefSet is greater than the minimum count of the least_diverse_sol and the HQRefSet. In case these two criteria are not satisfied the algorithm discards the output solution.

```
REFSET-UPDATE-METHOD(output_solution[], HQRefSet[], divRefSet[])
1.    if output_energy < HQ_worst_energy
2.        HQRefSet[HQWorstSolutionIndex] ← outputSolution
3.    else
4.        difference1 ← MINIMUM-DIFFERENCE(outputSolution, HQRefSet)
5.        difference2 ← MINIMUM-DIFFERENCE(least_diverse_sol, HQRefSet)
6.        if difference1 > difference2
7.            divRefSet[divWorstSolutionIndex]← outputSolution
```

Figure 4.2: Pseudocode for the *RefSet* Update Method

```
MINIMUM-DIFFERENCE(sol1[], sol_pool[])
1.    min_count ← sol1_size * 3 + 1
2.    for i ← 0 to sol_pool_size
3         sol2 ← sol_pool[i]
4.        count ← 0
5.        for j ← 0 to sol1_size
6.            amino_acid1 ← sol1[j]
7.            amino_acid2 ← sol2[j]
8.            for k ← 0 to 3
9.                difference ← abs(amino_acid1_torsion[k] –
10.                       amino_acid2_torsion[k])
11.               if difference > 5
12.                   count ++
13.       if count < min_count
14.           sol_min_count ← sol_pool[i]
15.           min_count ← count
16.   return min_count
```

Figure 4.3: Pseudocode for counting the minimum number of torsion angles that differ within $\pm 5^\circ$

The number of times MINIMUM-DIFFERENCE is called depends on how many output solutions we have. For each output solution, the running time of MINIMUM-DIFFERENCE is O(sol_pool_size * m).

The Termination criterion for our SS algorithm is based on the members of the *RefSet*. In case the *RefSet* is not updated with new solutions for ten consecutive iterations of the SS algorithm, the algorithm terminates.

4.6 Subset Generation Method

The Subset Generation Method in our approach is subset type-1, which has two elements in the subsets. For the first iteration of the SS algorithm, the number of subsets is $(b! / 2!(b-2)!)$, where b is the size of the *RefSet*. For this case the algorithm combines every possible pair of solutions. However, after the first iteration, we do not allow previously combined solutions to be combined again.

4.7 Solution Combination Method

Because none of the Combination Methods discussed in Section 3.1 apply to our context and solution representation, a Solution Combination Method is devised in an incremental method that incrementally adds the torsion angle values of the amino acids from one of the two target solutions, and calculates the partial energies of the partially constructed solution. That is, for each amino acid position in the combined solution the torsion angles from one of the target solutions are added and the energy function up to this position is calculated. The same scenario is implemented for the second target solution. The method keeps the angle values that give lower partial energy value. This procedure is repeated until the torsion angle values of all amino acids are specified.

The algorithm has at most $(b! / 2!(b-2)!)$ subsets of solutions to combine, where b is the size of the *RefSet*. For each of these subsets, there are $2m$ amino acid replacements, and for each replacement the energy function is calculated. Therefore, for each subset, the running time of the Combination Method is $O(2m\ (8m - 6))$, for all the subsets the running time is $O(b^2m^2)$. Thus the overall complexity of the algorithm for i iterations is $O(b^2m^2i)$.

Table 4.5 shows the updated *RefSet* of the second iteration for the 1CRN example after executing both the Solution Combination Method and the Improvement Method. Note that in the high quality *RefSet* nine out of ten solutions are changed from the initial *RefSet* in (Table 4.4) to better energy values, and seven out of ten in the diverse *RefSet* are changed to introduce more diversity in the set.

Table 4.5: *RefSet* after executing both the Solution Combination Method and the Improvement Method

Solution	Energy	Solution	Energy
1	9.23E+06	11	3.82E+12
2	9.57E+06	12	4.23E+12
3	9.64E+06	13	2.12E+13
4	9.67E+06	14	3.34E+13
5	9.95E+06	15	3.38E+13
6	2.53E+07	16	7.42E+13
7	4.25E+07	17	7.69E+13
8	6.34E+07	18	9.87E+13
9	8.80E+07	19	2.84E+15
10	6.98E+08	20	3.45E+15

Chapter 5 Empirical Results and Discussion

In this chapter, we evaluate the performance of our solution in generating native like structures for the backbones of our target proteins. We first describe the methodology of our experiments and present our experimental subjects. We use our algorithm to generate 3D protein structures and the results are then discussed.

5.1 Methodology

In our experiments, we use three proteins with known structures in PDB. Our reference PDB is that of the Brookhaven database (Bernstein *et al.*, 1977). Up to this point, there are 47509 known structures in this database.

We evaluate our results in two ways. In the first way, we consider the energy values over the iterations of the algorithm. For each updated *RefSet*, we find the minimum energy in that *RefSet*. In the second way, we evaluate the target protein's structural difference from the reference protein by calculating the RMSD of the solutions in the last *RefSet* to the reference protein.

RMSD is one method of evaluating the structure of a target protein. It calculates the average distance, expressed in Å, of the Cartesian coordinates of the atoms of the target and reference proteins. Our RMSD calculation is based on the Cartesian coordinates of the Cα atoms of the two proteins. The RMSD is given by (Carugo and Pongor, 2001):

$$RMSD = \frac{\sqrt{\sum_i (x_{ai} - x_{bi})^2 + (y_{ai} - y_{bi})^2 + (z_{ai} - z_{bi})^2}}{\sqrt{n}}$$

where a is the target protein, b is the reference protein, i is the position of an amino acid, and n is the total number of Cα atoms (i.e., the number of the amino acids of the protein).

The RMSD value of two identical structures is 0. It increases with the increase of structural difference among the two proteins. According to Carugo and Pongor (2001), the relationships between RMSD values and structural homology can be classified as follows:

- For RMSD values greater than 12Å, the two structures are completely unrelated.
- For RMSD values between 8 and 12Å, there is a questioned homology among the two structures.
- For RMSD values between 5 and 7Å, the two structures are related.
- For RMSD values between 1.5 and 4Å, the two structures are good match.
- For RMSD values between 0.6 and 1.4Å, the two structures are very good match.
- For RMSD values between 0 and 0.5Å, the two structures are almost identical.

The RMSD values of the last two points are more referenced by methods that use experimental techniques to generate 3D structures of proteins. For computational approaches higher values are more probable.

Our experimental subjects are 1CRN, Repressor of Primer (1ROP), and Uteroglobin (1UTG). 1CRN is rather small in size, 46 residues, and is mainly found in plant seeds. It is one of the most studied proteins both theoretically and experimentally, because its crystals diffract very well. Hence, experimental methods can easily generate 3D structures for 1CRN (Teeter and Hendrickson, 1979). 1ROPis a protein of size 56. It is found in a bacterium and its role is to regulate the number of copied genes in DNA molecules. Recent studies showed that in cancer cells the number of copied genes increases (Cappuzzo *et al.*, 2005). Thus, Repressor of Primer's function may be important cancer prevention studies. 1UTG is a protein of size 70. It is specific to rabbits, hares, and picas. Its precise role is still unknown (Dunkel *et al.*, 1995).

To the best of our knowledge, no computational method using our assumptions discussed in Section 4.1 has reported RMSD values for the structures of 1ROPand 1UTG. Regarding 1CRN, we are comparing our RMSD results with that of Schulze-Kremer (2000), which used the same assumptions as ours.

5.2 Experimental Results and Discussion

Figure 5.1 displays the graph of the minimum energy in each updated *RefSet* for 1CRN. Each updated *RefSet* is represented as an iteration number in the graph. This is because at each iteration solutions in the *RefSet* are updated. The x-axis represents iteration numbers and the y-axis represents energies in kcal/mol.

The minimum energy of the first *RefSet*, which is the *RefSet* constructed from the initial population, is relatively large since solutions in the initial *RefSet* are random solutions. After applying the SS operators on the solutions of the initial *RefSet*, the minimum energy in the second iteration decreases almost 70%. From iteration 2 onwards, the minimum energy decreases constantly until it reaches convergence. In all *RefSets*, the solution with minimum energy is a member of the high quality *RefSet*. In iterations 6 and 7 the minimum energy is not changed. That means no new solution with lower energy value than that of the one in iteration 5 is generated in both iterations 6 and 7. The same scenario happens with two sets of consecutive iterations: 9, 10, and 18, 19, 20. The algorithm terminates when there are no more new solutions entering the *RefSet* for 10 consecutive iterations. This criterion is depicted in Figure 5.1 from iteration 39 to 49. Prior to convergence, the decrease in the minimum energy from iteration to iteration is very small.

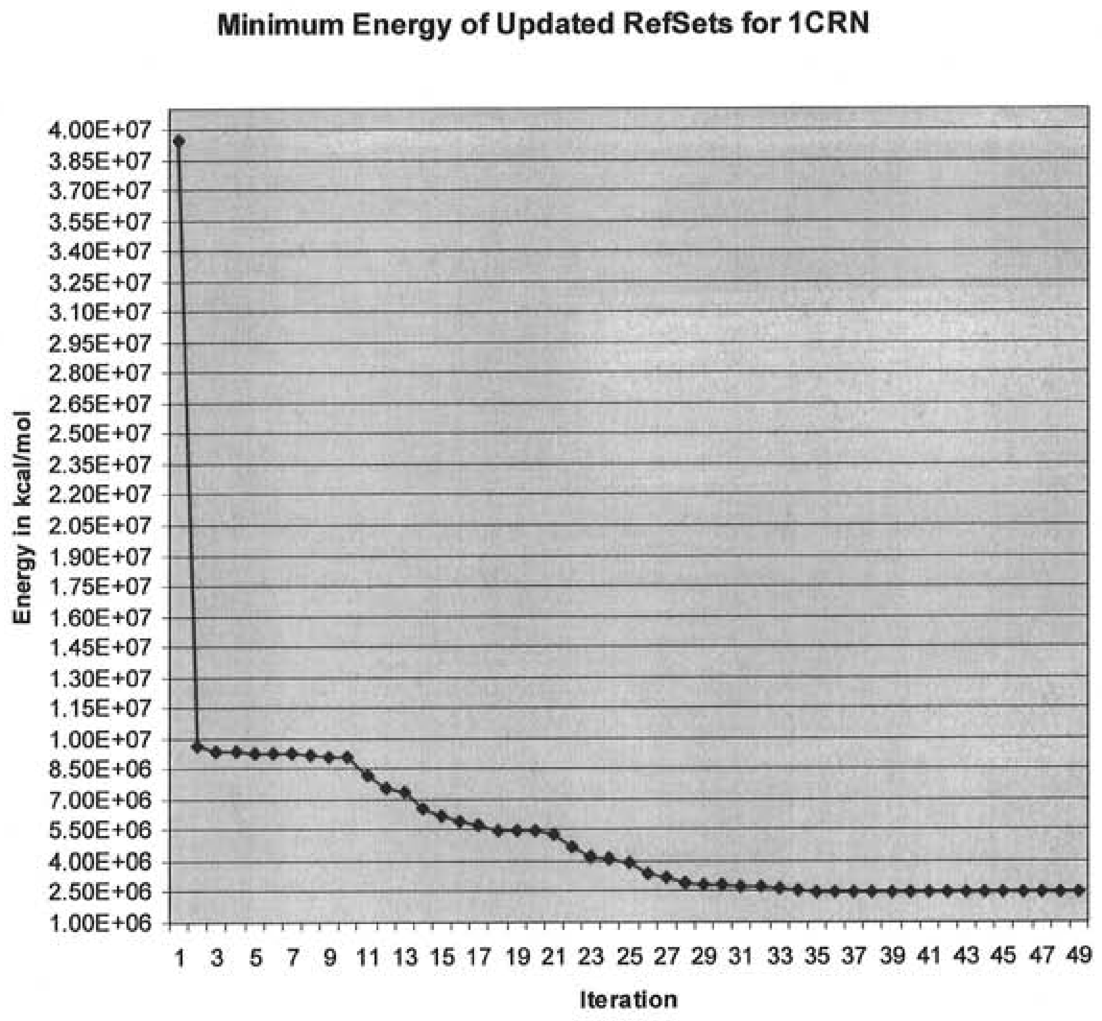

Figure 5.1: Minimum energy of each updated *RefSet* for 1CRN

We note that the energy of the reference 1CRN, with PDB torsion angles, according to our energy function is 2,519,470 kcal/mol. The minimum energy that we reached in the last iteration is 2,459,798 kcal/mol. This shows that our SS algorithm is reliable in reducing the energy value.

Table 5.1 displays the solutions in the last *RefSet* with their energy values and RMSDs to the reference 1CRN. The first ten solutions are the ones in the high quality *RefSet*, while the last ten solutions are the ones in the diverse *RefSet*. The minimum RMSD we reached is 9.43Å, which indicates a dubious similarity between the structure that our algorithm generated and the reference 1CRN. Note that this RMSD is not for the solution with minimum energy value. In addition, there is no accurate relationship between energy value and RMSD. The major problem lies in our energy function, which is not an accurate indicator of 3D structures of proteins. However, as we can see in solutions 11 to 20, we have relatively higher RMSD values for higher energy values. This proves that our energy function is working well with energies that differ in large amounts. Our minimum RMSD for 1CRN is comparable to that obtained by Schulze-Kremer (2000), which used a genetic algorithm and reached 9.15Å. The algorithm ran for roughly 9 hours.

Table 5.1: Potential energy values and RMSDs for 1CRN in the last *RefSet*

Solution	Energy	RMSD	Solution	Energy	RMSD
1	2459798	18.21	11	5.34E+12	18.68
2	2484300	19.12	12	3.98E+13	24.77
3	2492144	17.87	13	5.63E+13	18.69
4	2498973	19.98	14	8.97E+13	20.34
5	2534834	13.78	15	9.78E+13	23.31
6	2545990	16.67	16	3.26E+14	25.49
7	2549229	19.23	17	4.67E+14	25.49
8	2549839	10.23	18	8.87E+14	19.80
9	2549869	9.43	19	3.44E+15	26.48
10	2551423	13.89	20	5.80E+15	23.11

Figure 5.2 displays the minimum energy in each updated *RefSet* for Repressor of Primer. The algorithm ran for 78 iterations. The minimum energy it reached is 2897495 kcal/mol, while the energy of the reference 1ROPwith our energy function is 3021593 kcal/mol.

Table 5.2 shows the solutions in the last *RefSet* with their energy values and RMSDs to the reference Repressor of Primer.

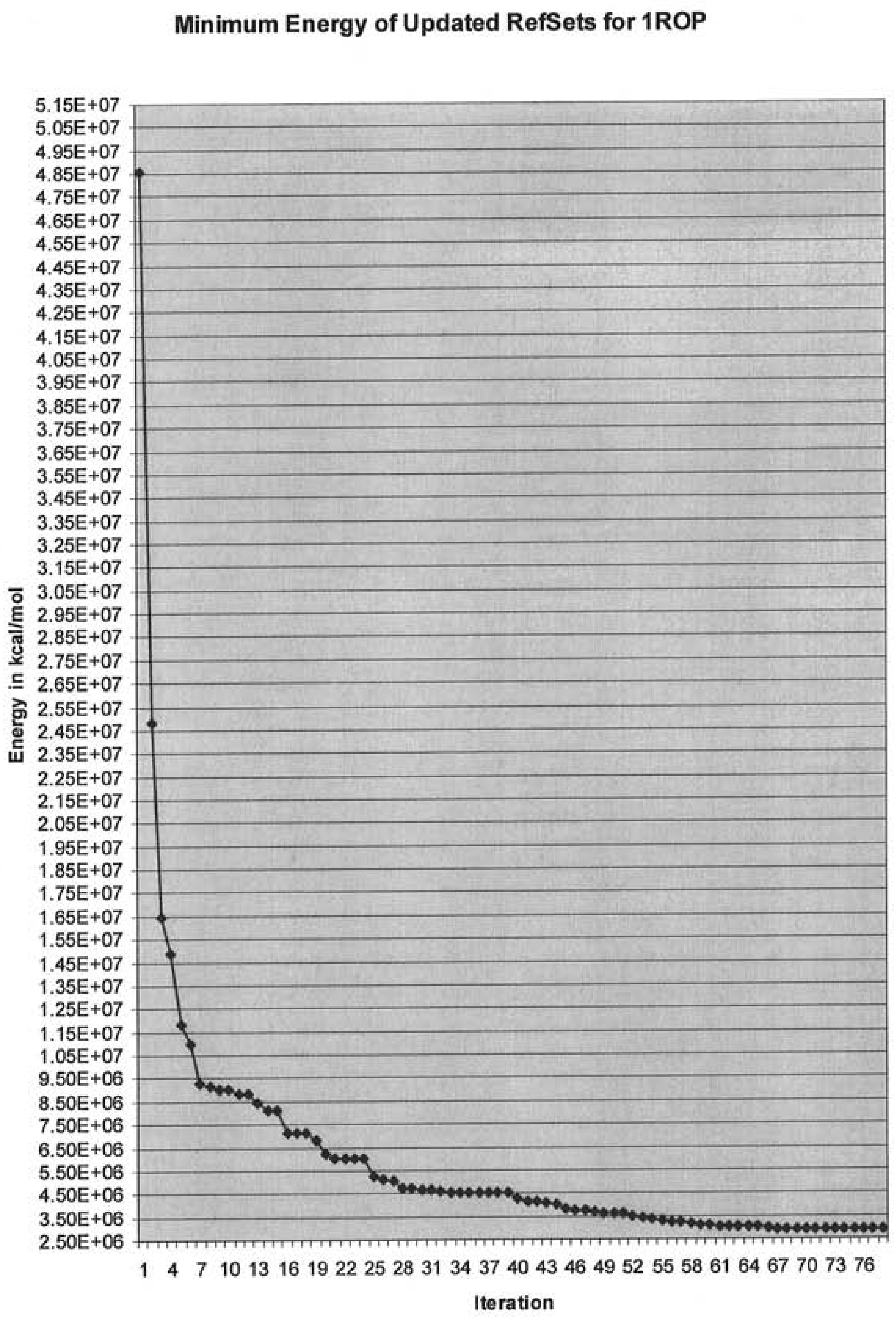

Figure 5.2: Minimum energy of each updated *RefSet* for 1ROP

The minimum RMSD of the produced solutions to the reference 1ROPin the last *RefSet* is 17.25Å. Note that the solution with minimum energy, solution number 1, is not the solution with minimum RMSD. Once again, this shows a discrepancy between the energy function representation of the protein structure and the RMSD; although the SS algorithm produces a solution with energy values less than that of the reference protein, this solution does not yield the lowest RMSD values. This discrepancy points to the inaccuracy of the energy function expression used. It also indicates that our assumptions, which eliminated some energy terms, have further decreased the accuracy of the energy function. Our choice of torsion angles is limited to the ten most occurring 10 degrees intervals in PDB for 100 proteins. This restricts the choice of angles, because throughout our entire algorithm we are selecting torsion angle values only from these ten intervals. In addition, ignoring the side chains in the computation of the energy function if another factor that has limited the quality of the solutions generated by our SS algorithm. However, for solutions with higher energy values, solutions 11 to 20, RMSD values are higher than the ones with low energy values, solutions 1 to 10. This shows again that although there is no accurate relationship between the energy values of solutions and their RMSD values, solutions with higher energy values lead to relatively higher RMSD values. The algorithm ran for roughly 15 hours.

Table 5.2: Potential energy values and RMSDs for 1ROP in the last *RefSet*

Solution	Energy	RMSD	Solution	Energy	RMSD
1	2897495	26.34	11	9.12E+11	31.43
2	2927498	27.12	12	2.10E+12	35.12
3	2946598	17.25	13	4.18E+12	32.53
4	3098678	21.51	14	6.91E+12	39.22
5	3147598	29.41	15	7.12E+13	30.24
6	3409180	18.14	16	9.81E+13	36.23
7	3580951	28.53	17	3.72E+14	41.23
8	3687235	28.21	18	8.79E+14	38.58
9	3693498	24.42	19	9.23E+14	38.26
10	3702349	19.12	20	3.24E+15	39.48

Figure 5.3 shows the minimum energy graph of each updated *RefSet* for 1UTG. The behavior of the minimum graph is the same as 1CRN and Repressor of Primer. The

algorithm ran for 111 iterations. The minimum energy we reached is 4001294 kcal/mol, while the energy of the reference 1UTG with PDB torsion angles is 4043153 kcal/mol.

Table 5.3 shows the solutions in the last *RefSet* with their energy values and RMSDs to the reference 1UTG. The minimum RMSD found is 20.63Å. For 1UTG the algorithm ran for roughly 24 hours.

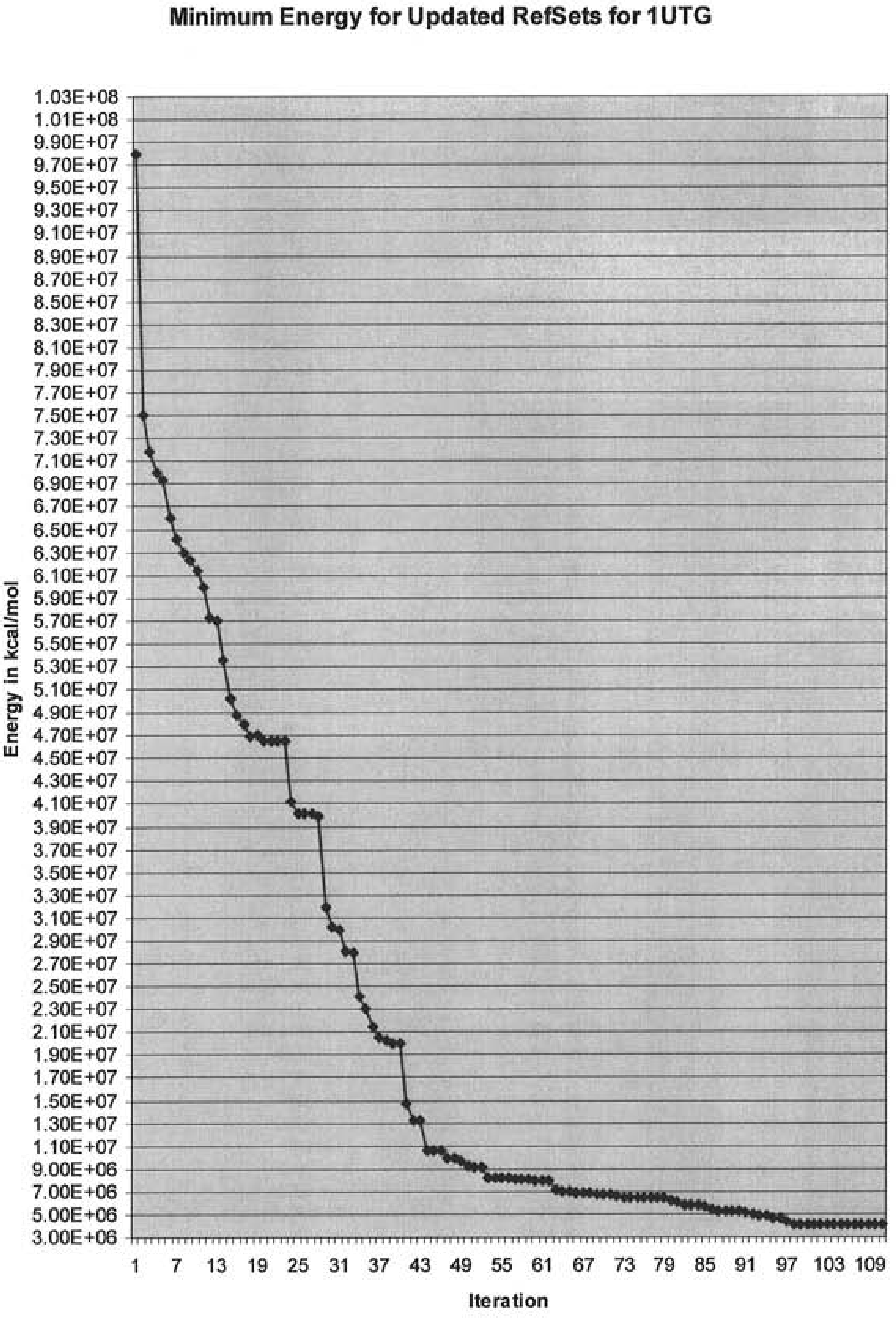

Figure 5.3: Minimum energy of each updated *RefSet* for 1UTG

Table 5.3: Potential energy values and RMSDs for 1UTG in the last *RefSet*

Solution	Energy	RMSD	Solution	Energy	RMSD
1	4001294	29.12	11	6.93E+12	35.45
2	4029489	31.42	12	7.12E+12	37.34
3	4099384	21.42	13	4.52E+13	39.14
4	4137499	30.54	14	4.91E+13	33.54
5	4198986	28.31	15	5.19E+13	39.65
6	4212939	27.44	16	2.12E+14	37.53
7	4279128	20.63	17	6.05E+14	38.65
8	4399183	39.72	18	3.92E+15	38.87
9	4398724	23.45	19	9.24E+15	39.65
10	4361238	26.64	20	2.06E+16	37.88

We can infer that our SS algorithm is an excellent algorithm for finding low energies. Moreover, we can say that there are some limitations in our approach that we believe are the reasons for producing relatively high RMSD values.

Chapter 6 Conclusion and Future Work

We have presented a scatter search algorithm for building 3D structures of proteins. Given an amino acid sequence, our algorithm generates 3D structures for the backbone of this sequence. Each generated structure is represented as a solution in a population. A structure is characterized by the backbone torsion angles, phi, psi, and omega of the protein. The algorithm starts by generating random solutions in the initial population. However, controlled randomization is employed throughout the algorithm by restricting the choice of the torsion angles to protein data bank data. Solutions are then evaluated to be part of the *RefSet*, which in all the iterations of the algorithm holds two types of solutions: high quality and diverse. A solution is evaluated by its potential energy. High quality solutions have low energy values. Thus, our algorithm's main focus is to decrease the energy values of the solutions. Moreover, the energy function is manipulated by the torsion angles, thus, SS operators are applied on the torsion angles of the solutions. We have two SS operators that modify the solutions in search of better solutions, these operators are: the Improvement Method and the Combination Method. Solutions in the *RefSet* are first combined to yield new solutions that are later improved. An improved solution is then evaluated to be a member of the *RefSet* in the next iteration. This process is repeated until there are no more new solutions to be part of the *RefSet* for ten consecutive iterations.

We evaluated our algorithm on three proteins taken from a PDB in two ways. In the first way, we showed the energy decrease over the algorithm. In the second way, we calculated the structural difference of the protein our algorithm generated with that of the one in PDB. We used RMSD as a measure of similarity between the structures. For the three proteins, our algorithm is able to generate structures with energies less than that of the reference proteins. Moreover, for proteins with relatively small in size, our algorithm generates structures with RMSD values that are comparable to that of existing approaches. For example, the RMSD value of 1CRN, 46 residues, we reached is 9.43Å.

This value is comparable to an existing approach that used genetic algorithm and reached 9.15Å. For Repressor of Primer, 56 residues, and 1UTG, 70 residues, our algorithm generates structures with 17.25Å, and 20.63Å RMSD values, respectively. The major problem lies in our energy function, which is not a real indicator of the native 3D structures of the proteins. Many energy models, mathematical, experimental, or statistical are still not reliable enough to differentiate native or non-native structures. Moreover, the assumptions that we used in order to cut the cost of the execution time further decreased the accuracy of the energy function. However, SS approach proved to be a reliable search algorithm for the PSP problem, since it exhaustively searches the possible search space of proteins' structures to find structures with low energy values.

There are some promising future directions that can be integrated to our algorithm. These directions are mainly the implementation of the assumptions that we made. We assumed constant binding geometry; by keeping all bond lengths and bond angles constant we are ignoring the real behavior of proteins, where bond lengths and bond angles vary with a small variation in any of the torsion angles. This factor leads to a simplified version of the CHARMM energy model, where bonds, lengths, and improper torsions components of the CHARMM energy model are canceled. In addition, our energy potential function can be expanded to include hydrogen bonding, which calculates the forces exerted by hydrogen bonds among hydrogen acceptors and hydrogen donors. In this case, the H attached to amino N of every amino acid should be added in order to calculate the hydrogen component of the energy function.

Our solution completely discardes the side chains associated with every amino acid backbone. This is a great simplification in a protein representation, since side chains are the groups that differentiate amino acids from each other. In addition, side chains largely contribute to the folding process of proteins. By adding side chains in future works, more realistic protein structures can be generated.

Another factor that is ignored in our approach is that of the surrounding reactions with – or solvent energy contribution to – the target protein. In future works, reactions with the solvents can be added to the energy function in order to simulate the real folding process. This has been recently addressed in CHARMM22 version ((Brooks *et al.*, 2006).

References

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990). A basic local alignment search tool. *Journal of Molecular Biology*, 215, 403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman D.J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25, 3389-3402

Bowie, J.U., Luthy, R. and Eisenberg, D. (1991). A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 253(5016), 164-170.

Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S. and Karplus, M. (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4(2), 187-217.

Brooks, C.L., Chen, J. and Im, W. (2006). Balancing salvation and intermolecular interactions: toward a consistent generalized born force field (CMAP opt. for GBSW). *Journal of American Chemical Society*, 128, 3728-3736.

Bui, T.N. and Sundarraj, G. (2005). An efficient genetic algorithm for predicting protein tertiary structures in the 2D HP model. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, 385-392.

Cappuzzo, F., Hirsch, F.R., Rossi, E., Bartolini, S., Ceresoli, G.L., Bemis, L., Haney, J., Witta, S., Danenberg, K., Domenichini, I., Ludovini, V., Magrini, E., Gregorc, V., Doglioni, C., Sidoni, A., Tonato, M., Franklin, W.A., Crino, L., Bunn, P.A., Varella-Garcia, M. (2005). Epidermal growth factor receptor gene and protein and gefitinib sensitivity in non-small-cell lung cancer. *Journal of National CancerInstitute*, 97, 643-655.

Carugo, O. and Pongor, S. (2001). A normalized root-mean-square distance for comparing protein three-dimensional structures. *Protein Science*, 10, 1470-1473.

Chou, P.Y. and Fasman, G.D. (1974). Prediction of protein conformation. *Biochemistry*, 13(2), 222-245.

Cui, Y., Chen, R.S. and Wong, W.H. (1998). Protein folding simulation with genetic algorithm and supersecondary structure constraints. *PROTEINS: Structure, Function, and Genetics*, 31, 247-257.

Dunkel, R., Vriend, G., Beato, M. and Suske, G. (1995). Progesterone binding to uteroglobin: two alternative orientations of the ligand. *Protein Engineering*, 8, 71-79.

Forman, S.L. (2001). *Torsion angle selection and emergent non-local secondary structure in protein structure prediction*. PhD thesis, The University of Iowa, Iowa City, IA.

Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability: a guide to the theory of NP-completeness*. New York: Freeman.

Glover, F. (1998). A Template for Scatter Search and Path Relinking. In J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), *Artificial Evolution*, Lecture Notes in Computer Science 1363, Springer-Verlag, 13-54.

Gribskov, M., McLachlan, A.D. and Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 84, 4355-4358.

Jones, D.T., Taylor, W.R. and Thornton, J.M. (1992). A new approach to protein fold recognition. *Nature*, 358(6381), 86-89.

Jones, D.T. (1998). THREADER: protein sequence threading by double dynamic programming. In *Computational Methods in Biology* (ed. Salzberg, S., Searl, D. and Kasif, S.). Amsterdam: Elsevier Science.

Jorgensen, W.L., Maxwell, D.S. and Tirado-Rives, J. (1996). Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *Journal of American Chemical Society*, 118, 11225-11236.

Klepeis, J.L. and Floudas, C.A. (2003). *Ab initio* tertiary structure prediction of proteins. *Journal of Global Optimization*, 25, 113–140.

Kopp, J. and Schwede, T. (2004). Automated protein structure homology modeling: a progress report. *Pharmacogenomics Journal*, 5(4), 405-416.

Krogh, A., Brown, M., Mian, I.S., Sjolander, K. and Haussler, D. (1996). Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235, 1501-1531.

Laguna, M. and Marti, R. (2003). *Scatter search: Methodology and implementations in C*. Boston: Kluwer Academic Publishers.

Lathrop, R.H. (1994). The protein threading problem with sequence amino-acid interaction preferences in NP-complete. *Protein Engineering*, 7(9), 1059-1068.

Notredame, C. (2002). Recent progress in multiple sequence alignment: a survey. Pharmacogenomics Journal, 3(1), 131-144.

Pandit, S.B., Zhang, Y. and Skolnick, J. (2006). Tasser-lite: an automated tool for protein comparative modeling. *Biophysics Journal*, 91(11), 4180-4190.

Parsons, J.R., Holmes, J.B., Rojas, J.M., Tsai, J. and Strauss, C.E. (2005). Practical conversion from torsion space to Cartesian space for In Silico protein Synthesis. *Journal of Computational Chemistry*, 26(10), 1063-1068.

Prusiner, S.B. (1998). Prions. *Proceedings of the National Academy of Sciences of the United States of America*, *95*, 13363-13383.

Przybylski, D. and Rost, B. (2004). Improving fold recognition without folds. *Journal of Molecular Biology*, 341, 255-269.

Ramachandran, G.N. and Sasiskharan, V. (1968). Conformation of polypeptides and proteins. *Advances in Protein Chemistry*, 23, 283-437.

Schulze-Kremer, S. (2000). Genetic algorithms and protein folding. *Methods in Molecular Biology*, 143, 175-222.

Setubal, J. and Meidanis, J. (1997). *Introduction to computational molecular biology*. Boston: PWS Publishing Company.

Sikder, A.R. and Zomaya, A.Y. (2005). An overview of protein-folding techniques: issues and perspectives. *International Journal of Bioinformatics Research and Applications*, 1(1), 121-143.

Skolnick, J., Kihara, D. and Zhang, Y. (2004). Development and large scale benchmark testing of the PROSPECTOR 3 threading algorithm. *Proteins*, 56, 502-518.

Smith, T.F. (1995). Letter. *Science*, 268, 958-959.

Teeter, M.M. and Hendrickson, W.A., (1979). Well ordered crystals of the plant seed protein 1CRN. *Journal of Molecular Biology*, 127, 219-224.

Unger, R. and Moult, J. (1993). Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231, 75-81.