CONSTRAINT APPROACH FOR PROTEIN STRUCTURE PREDICTION IN THE SIDE CHAIN HP MODEL

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
ALBERT-LUDWIGS UNIVERSITY OF FREIBURG
JUNE 2008

Done by: Mohamad Rabbath
Born on: 01.01.1983

Under the chair and main supervision of: Prof. Dr. Rolf Backofen

Direct Supervisors: Dr. Sebastian Will
Martin Mann

Table of Contents

Abstract

Protein structure prediction has been always a very interesting problem to solve, especially in the last 10 years. Many previous methods tried to focus on HP model prediction, however most of those methods have the drawback of giving approximate solutions (like [14]). Another drawback of most of those works is that they ignore the representation of the side chains of the amino acids [27].

In this master thesis, we develop an approach of a concrete constraint model that takes the side chains of the amino acids into consideration and gives the exact solutions for a given sequence in the side chain HP model in terms of minimizing the energy, without any approximation.

In this work we also present the obtained results of the predication model and show some important statistics especially about the degeneracy. In addition to that we present some interesting results on generating protein like sequences, although this is a hard task in the model. The protein like sequences can be found in a very low probability in random sequences in the side chain model as we explain in this thesis.

Acknowledgements

I would like to thank Prof. Dr. Rolf Backofen the head of Bioinformatics team in the University Of Freiburg, who motivated me through his impressive lectures and publications, to do my master thesis under his chair. Also I want to thank him for allowing us through several events and seminars to share the ideas between all the team, and see the interesting problems and algorithms that other members work on.

I am very thankful for my supervisor Dr. Sebastian Will, who I knew a year ago through his course in Constraint Programing, and it was very impressive and motivating. I have to thank him for all his efforts during my master thesis to help me with new ideas and algorithms.

Also I want to thank my second supervisor Martin Mann. I am very grateful for his persistent help during my thesis, in answering my questions, giving me new ideas, and his developed libraries are very impressive, well designed and easy to integrate.

Finally i have to thank my parents; my mother Hala, and my father Dr. Haitham who raised me in a liberal way and encouraged me always for critical thinking. I want to thank all the people and friends who helped me during my life, and I apologize for not being able to mention all of them.

Freiburg, 19.06.2008,

Mohamad Rabbath

Chapter 1

Introduction

1.1 Motivation

Understanding the protein structure is playing a big role in the modern science, because the protein is the main block of any living being; and knowing the structure of the protein allows us to get into its functionality (The structure of the protein creates its functionality).Till today the X-ray cystography and nuclear magmatic resonance (NMR) spectroscopy are still difficult, expensive and time consuming to apply. Therefore the traditional biochemical methods of predicting the protein structure are not as fast as the growing knowledge of new proteins sequences, and this can be recognized in the proteins structures that are deposited in the public database PDB (Protein Data Bank of experimentally determined 3D structures of proteins)[9]. Therefore we are going to give an exact prediction method depending on some assumptions that imply some simplicities in the model and allow us to add a new step on understanding the complex exact structure of the protein, which is still very hard task, and even simplified protein variants were shown to be NP-Complete [5],[15].In this thesis we are trying to give an exact 3D prediction of the protein in two kinds of lattices (Cubic and Face Centered Cubic as we will see later in the introduction) and based on one property of the protein (The H or P property, as we will also see in the next section of the introduction), and based on the assumption that each amino acid in the sequence is not a simple monomer, but rather a dipole consisting of a backbone and a side chain, which is biologically more realistic. Figure (figure 1.2) shows us that each amino acid is connected to the next amino acid in the sequence through a peptide bone which connects the carboxyl group of the first amino acid to the amino

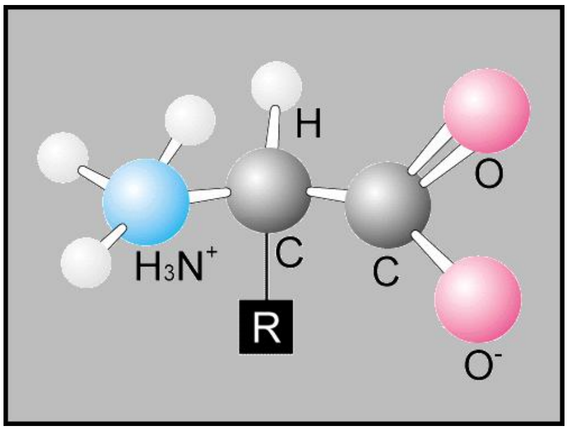


Figure 1.1: All amino acids contain 3 parts, an amino group (NH3+),a carboxyl group (COO-)and an "R" group. The "R" group varies among amino acids. This image is taken from http://www.rothamsted.ac.uk/notebook/courses/guide/prot.htm

group of the second amino acid. The main block of the protein is the amino acid, and each protein consists of a sequence of amino acids. The general structure of the amino acid is shown in figure (1.1). All the amino acids have the same main general structure that is formed by the amino group (NH2) and carboxy group (COOH) while differs only in the R part which is connected to the first C atom that is called α carbon atom. The chemical group R (the side chain of the amino acid) gives the protein its unique properties such as being hydrophobic or hydrophilic, small or large, charged or uncharged. And in this thesis we will focus only on one property which is hydrophobicity or polarity .

1.2 Overview of the Protein Representation in HP Model

Since the aim of the work is to formally predict the structure of the protein, the protein should be formally represented. There are many ways for protein representation, and all of them depend on the chemical properties of the protein which are decided by its side chain (hydrophobic or hydrophilic, small or large, charged or uncharged). One of the most common representation of the protein is the HP Model which introduced by Dill ([7]). The hydrophobicity is not only the most important physico-chemical

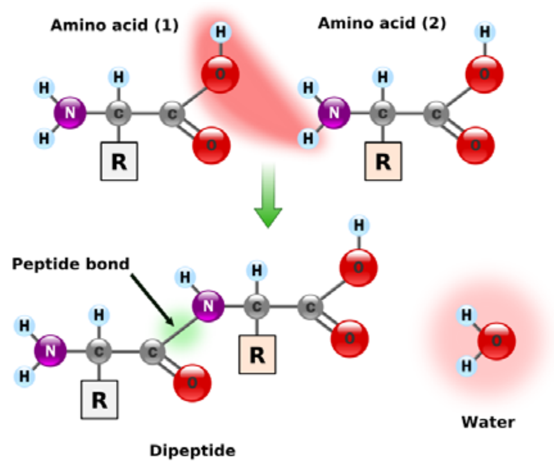


Figure 1.2: Two amino acids connected by a peptide bond. This image is taken from the wikipedia.

characteristic of amino acids, but also very poorly defined. The word hydrophobic and hydrophilic literally means "afraid of water" and "fond of water". The hydrophobic amino acids tend to cluster together and as a result from repelling from a mass of water .The residues classified as hydrophobic prefer to be in a non-aqueous environment, whereas the hydrophilic (polar) residues like to lay at the outside of water soluble proteins. The HP model allows the simplicity of reducing the amino acids alphabet to the set h,p, and this simplicity allows a later well constrained model. HP model was studied many times with many models, but most of them were concentrating of a simple representation of the amino acid as a single monomer. There are also several ways of representing the protein in the HP Model in Lattices such as the Cubic lattice and FCC (Face Centered Cubic) lattice which we are going to describe in next chapter of preliminaries and fundamental concepts.

1.3 Contribution

Our main contribution in this master thesis is to introduce a constraint satisfaction approach to predict the structure of the protein in side chain HP model. This will guarantee both the reasonable biological assumption of taking the side chain into consideration (as well as the backbone), and at the same time we get exact solutions rather than any approximation. The lattices representing the protein in this approach will be both the Cubic lattice and the FCC (which are the most common ways of 2D representation). The thesis will also give general statistics and results of the high degeneracy resulted from this model, and compare it to that obtained in the model without side chain [27].

1.4 Related Works

Indeed there were many articles from late 80s and eraly 90s tyring to explain how to decrease the gap existed in the database between the known sequences and their strucures (reviews [20],[21]- books [8],[24]). Also many works investigated the kinetics of the protein folding using lattice protein modles such that [23], [2],[4]. Such works require the prerequisite of knowing the optimal structures for the sequences, but till now most of the protein prediction works whether in 3D lattices [16],[27],[6] , or 2D lattices [17],[1],[25] do not consider the side chain representation although biologically it will be more accurate. Some of them even try to approximate the solution by simulating the folding process and/or applying statistical approaches [25],[16],[17],[1]. For example the work of [2] uses the Monte Carlo algorithm to simulate the folding mechanism of the protein claiming that folding begins with rapid collapse but then it is followed by slow search through stable core which is semi-compact and sequence-dependent. Even the methods that take the side chain into consideration like [10], still have the drawback of not giving exact solutions (but rather approximations). In our approach however we will try to give both an exact solution based on constraint satisfaction approach of minimizing the energy and with the consideration of representing the side chain in 3D Lattices (both Cubic Lattice and FCC (Faced Centered Cubic)).

Chapter 2

Preliminaries and Fundamental Concepts

In this chapter we will shortly give a small background and fundamental definitions needed in this thesis, as well as we will give and idea about representing the protein in side chain HP model in 2D and explain some property resulted, before going through our model in the next chapter.

2.1 Lattices

In this section we will define the lattice as the main mathematical object to represent the protein structure.

Definition 2.1.1. A lattice L is a set of lattice vectors such that:

$$\vec{0} \in \vec{L} \tag{2.1.1}$$

$$\vec{u}, \vec{v} \in \vec{L} \Rightarrow \vec{u} + \vec{v} \in \vec{L}, \vec{u} - \vec{v} \in \vec{L} \tag{2.1.2}$$

where of course the $+$, and $-$ are the addition, subtraction operations in the vector group and $\vec{0}$ is the zero vector.

There are many kind of lattices both for 2D representation and 3D, and we will describe 3 kinds of lattices namely the square lattice, cubic lattice and fcc lattice.

2.1.1 Square Lattice

The square lattice is the simplest well known 2D lattice. it is defined by its basis

$$\{(0,1),(1,0)\}$$

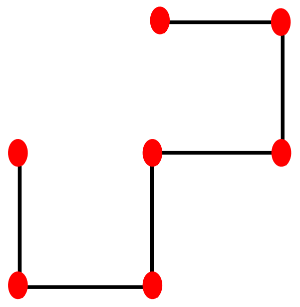


Figure 2.1: In this figure you can see an example of a square lattice that includes some monomers (regardless they are side chains or backbones, here it is just a general figure).

The minimal vectors in this lattice can be given as

$$\{(0, \pm 1), (\pm 1, 0)\}$$

An example of the square lattice can be seen in figure (2.1).

2.1.2 Cubic Lattice

The cubic lattice is also commonly used in the 3D representation in many applications (protein structure is one of it). The basis representing this lattice is

$$\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$$

The minimal vectors in the cubic lattice are represented in the set

$$\{(0, 0, \pm 1), (0, \pm 1, 0), (\pm 1, 0, 0)\}$$

An example of the cubic lattice can be seen in the upper part of figure (2.2).

2.1.3 Face Center Cubic (FCC) Lattice

The FCC is another lattice that forms in many cases more realistic way of representation in 3D than the cubic lattice. The basis of this lattice is the set

$$\{(0,1,1),(1,0,1),(1,1,0)\}$$

The minimal vectors set of the fcc lattice is

$$\{(0,\pm1,\pm1),(\pm1,0,\pm1),(\pm1,\pm1,0)\}$$

The FCC lattice can be seen in the lower part of figure (2.2).

2.2 Protein Structure in Side Chain

Since we want to use some lattice landscape to represent the protein structure we need to give a definition for the structure of the protein in the side chain model. Of course unlike the structure in the case of back bone one monomer representation, we have here to represent each amino acid as two monomers, one for the side chain and another for the backbone, and therefore we define the structure as following:

Definition 2.2.1. A structure str of a sequence seq is denoted as a tuple $str \in L^{|seq|} \times L^{|seq|}$, where L is a lattice, such that each element str_i has a backbone and a side chain and will be denoted resp. $str_i.b, str_i.sch$,(This way of denoting is close to programming languages) and should match the following conditions :

1. $\forall 1 \leq i \leq |seq| : str_i.b$ and $str_i.sch$ are neighbors.
2. $\forall 1 \leq i < |seq| : str_i.b$ and $str_{i+1}.b$ are neighbors.
3. $\forall i \neq j : str_i.sch \neq str_j.sch$ & $str_i.b \neq str_j.b$ & $str_i.sch \neq str_j.b$

Further more we need to define the surface of set of points P, because some times it is easier to deal with the surface than dealing with the set of points repressing the structure.

Definition 2.2.2. $Surface(P) = |\{(p, \grave{p}), p \in P$ and $\grave{p} \notin P\}|$

We also call each tuple $p, \grave{p}$ where p in P and $\grave{p} \notin P$ as surface pair. The previous definition for the surface means simply the number of surface pairs for a set P. An example of surface pairs can be seen in figure 2.3.

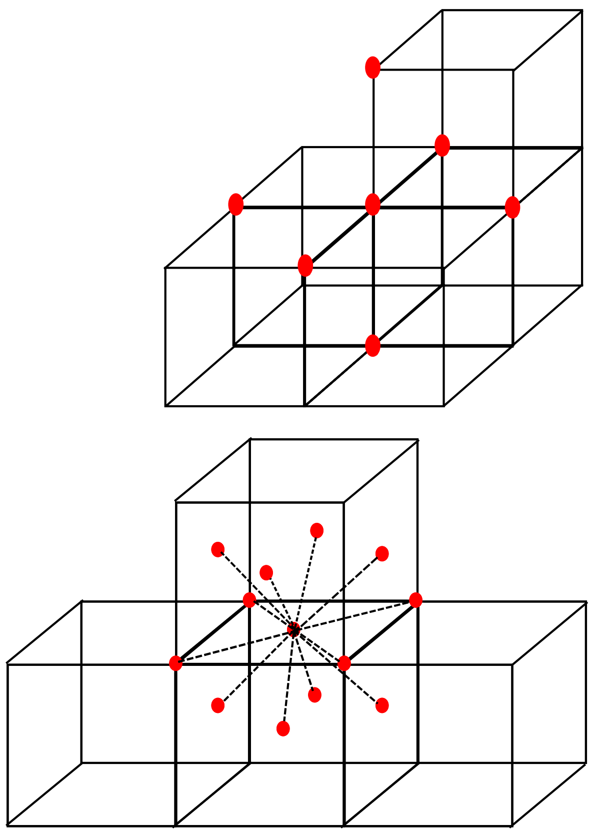


Figure 2.2: In this figure you can see an example of a cubic lattice that includes some monomers (regardless they are side chains or backbones, here it is just a general figure).

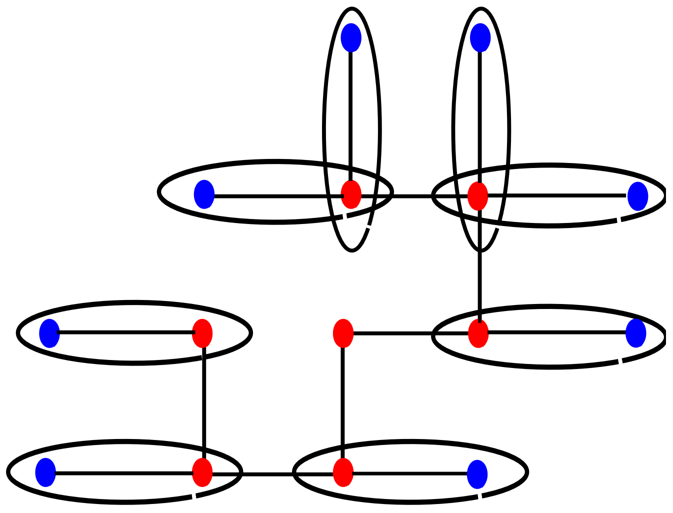


Figure 2.3: This figure shows example of the surface pairs (only subset of the surface pairs) of a set P, where only the red points belong to P.

2.3 Constraint Programming and Gecode

We promised in our thesis to give an exact constraint approach for our model, and therefore we have to define the Constraint Satisfaction Problems.

Definition 2.3.1. Constraint Satisfaction Problems (CSPs) are the set of mathematical problems that we must find states of objects that satisfies our constraints.

The CPSs are interesting problems in Computer Science especially in Artificial Intelligence. Many CSPs require heuristic and advanced method to implement in reasonable time. The Gecode c++ library is one of the successful implementation that offers variety of CSPs propagators (such the all different propagator, linear propagator, ...) in efficient implementation. Gecode library is an open source library and can be downloaded for free from http://www.gecode.org under the GECODE LICENSE AGREEMENT. Our constraint approaches used the Gecode for many constraint implementations, and it showed successful results. For now we have all the definitions we need to talk about some properties of the side chain optimal Hydrophobic core in a 2D square lattice, and the reason we chose it because we will not study the 2D representation in this thesis, but we will explain that the optimal hydrophobic core is unique in the 2D structure for any number of points.

2.4 Side Chain-Backbone Representation of the Protein

In this representation we are interested in HP protein model where each amino acid is not represented as a simple monomer, but rather as a side chain which is connected to a backbone. So each protein structure will be represented as connections of tuples of monomers. The backbones are connecting the structure through the neighboring relationship.We assume that both the side chain and the backbone have the same size, and we consider the approximation which assumes that the distance between a side chain and its backbone is equal to the distance between the backbone and the neighboring backbone, and equals the length of one bond in the lattice. Each monomer whether it is a backbone or a side chain should be restricted to the lattice. For this side chain-back bone model the same kind of lattices are used as those used in the -one monomer for one amino acid- representation. Here in this section we are going to talk about the definition of the Hydrophobic core, and about some properties of the side chain model in 2D square lattice, while in the next chapter we begin talking about protein prediction in 3D, and we explain how to model our problem.

2.4.1 The Concept of the Hydrophobic Core in the Side Chain Model

The concept of the Hydrophobic core were used in several literatures and studies in the protein HP model with and without side chains (ex. [7],[26]).The Hydrophobic core definition in the HP mode with the side chain can be shown below.

Definition 2.4.1. The hydrophobic core in a structure in side chain model is the set of positions occupied by the side chains of Hs amino acids in the structure.

2.4.2 Some Properties of the Side Chain Model in 2D Square Lattice

It turned out that the optimal solution of the hydrophobic core of the square lattice has a specific shape. Our study here for the property considers only the dipoles package without applying the neighboring constraints of the backbones .In case both we have even or odd number of amino acids we will get a unique optimal solution (in case the number is odd we will get four symmetrical solutions), figure 2.4.

We will give a proof for such a structure. But before giving it let's consider the following notes:

1. **Note 1:** For any given finite number of monomers in a square lattice (and generally in any arrangement) we have $4n = 2c + s$, where n is the number of monomers and c is the number of contacts between the side chains and s is the surface.

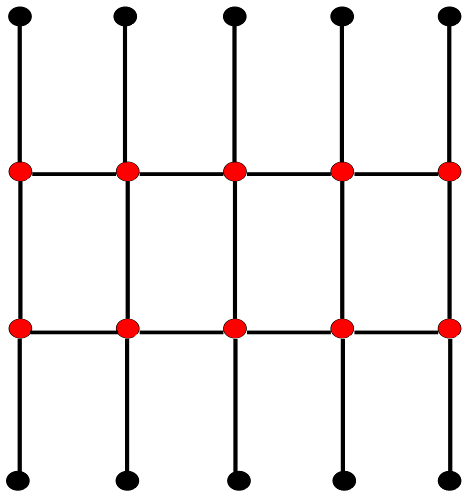
2. **Note 2:** In any given 2 dimensional finite square lattice, we have the relation $s = n + boundPoints$, where n is the number of monomers,s is the surface and $boundPoints$ are the number of boundary points, where we define the boundary point as following:

   Definition 2.4.2. We say that a point $p_i \in Str$ is a boundary point, where Str is a connected structure in a square lattice iff this point has exactly two neighboring points which does not belong to the structure Str

   An example of boundary points could be seen in figure 2.5.
   And since the minimum number of boundary points could be 4 so the minimum surface side is: $s = n + 4$

3. **Note 3:** From Note1,Not2 we can see the maximum number of contacts $maxC$ that can be obtained in square lattice is $maxC = (4n - n - 4)/2 = 3n/2 - 2$.
   Now we can state our lemma.

Lemma 2.4.1. *The only optimal HCore structure (or the optimal HP solution for a protein with only hydrophobic sequence of amino acids) in a square lattice is two parallel neighboring lines filled by the side chain neighboring monomers with the backbones are connecting to the outside (figure 2.4).*



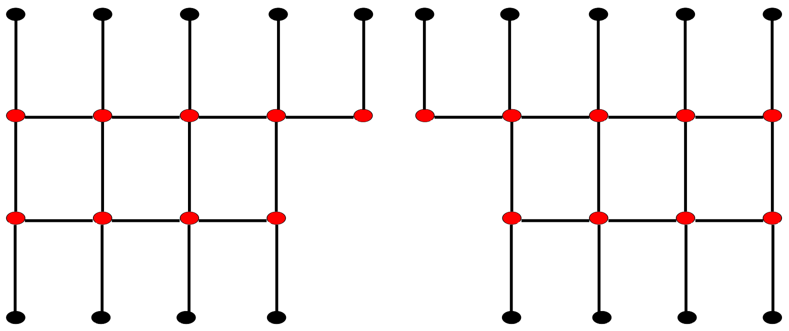


Figure 2.4: In this figure we see the optimal solution for the hydrophobic core in the square lattice. Above we have identical solution for even dipoles, and in the button also we have one solution (but with four symmetrical cases -two of them are shown-, in case we have odd number of amino acids).

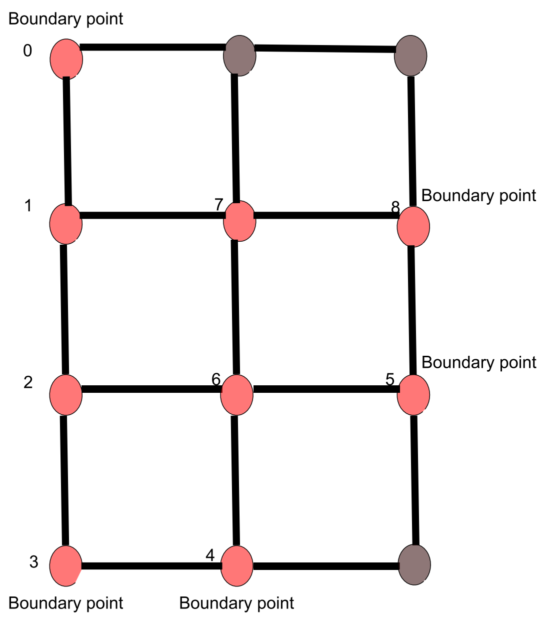


Figure 2.5: In this figure you can see the ordering coding of the points. The points that belongs to the structure are the pink points.

Proof. **Proof for the first part -The structure shown in figure 2.4 is an optimal solution-:** According to Note3 the optimal number of contacts should be maximally (3n/2-2) which means with even number of dipoles it is 3n/2-2 while with odd number of dipoles it is 3(n-1)/2-2 because the number of contacts should be natural of course. we prove that the structure in the figure already has this optimal number of contacts by induction. For even numbers:
n=2 obviously we have 1=3*2/2-2 contact.
We suppose that our structure has 3n/2-2 for n monomers, then for (n+2) monomers (remember , now we are inducting over the even numbers) we have according to the structure after adding 2 more monomers 3 more contacts whether we add them on the left or the right. Thus, we have now 3n/2-2+3=3(n+2)/2-2 contacts so , therefore the relation holds for the even number of monomers. We use the same way for odd numbers. n=1 obviously we have 0=3*(1-1)/2-2 contacts.
Now we suppose our structure has 3(n-1)/2-2 contacts for our n odd monomers. Now adding 2 more monomers will add also 3 more contacts, so now we have 3(n-1)/2-2+3=3(n+1)/2-2 contacts and again the relation holds for odd n+2 number of dipoles, Thus the rehalation holds. □

Sketch Proof for the second part -If a structure is an optimal solution, then it should be the shown in figure 2.4-: Here we give a sketch idea of the reason and not a full proof. First of all we have to notice that: the upper left point, the upper right point, the bottom left point, and the bottom right point; each of wich has maximally 2- contacts to another side chains in any finite square lattice,because as usual the other contact should be lost for the back bone and another one will be lost according to the landscape of the cubic lattice (the dashed line), as shown in figure 2.6. Now it is obvious that by maximizing the number of contacts each side chain achieve we will maximize the global H-H number of side chains contacts. The maximum number of contacts can be achieved for any internal side chain will maximally achieve 3 contacts to other side chains like figure 2.7. It is easy to prove by construction that the solution that maximize the internal contacts to 3 is unique, and it matches the shape resulted in figure 2.4, as well as the corner monomers have their maximally 2 contacts. So since this shape that maximize the contacts obtained for each individual is unique, and any other shape will let at least one dipole loosing a contact, so the global maximum contacts for the side chains of a core are only obtained by this unique shape.

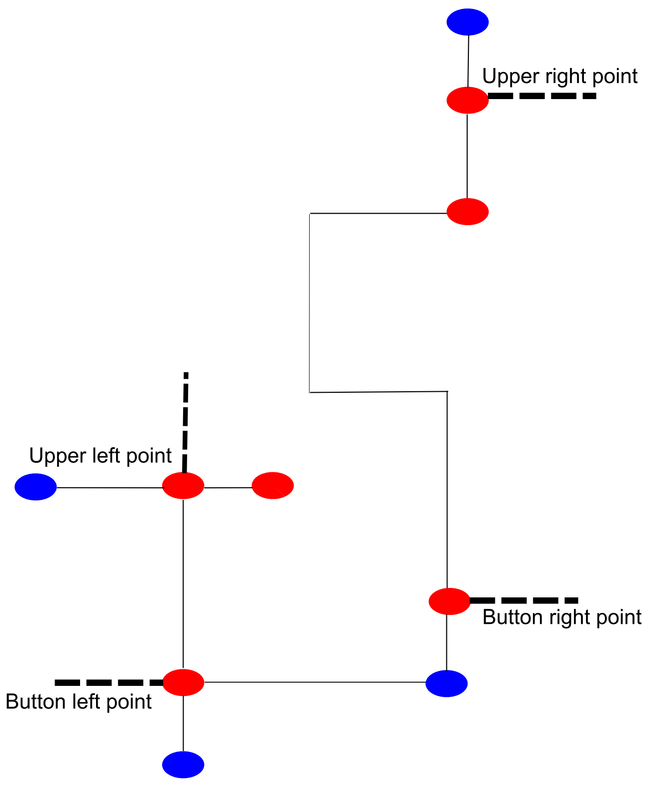


Figure 2.6: In this figure the back bone of each extreme point was colored in blue, and you can notice that each point in the corner of a finite cubic lattice will loose at least two side chains contacts, one for the backbone as usual and the other because of the nature of the landscape of cubic lattice.

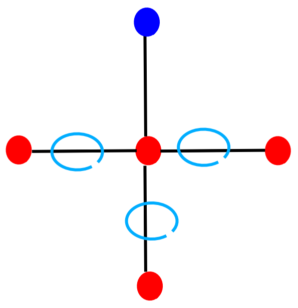


Figure 2.7: In this figure we can see the maximally obtained side chains contacts for any internal dipole. The red monomers are side chains while the blue is a backbone. By construction this maximally obtained internal contacts are unique.

Chapter 3

From Naive to Concrete Side Chain Model

In this chapter we are going to talk about the first naive approach for the side chain model, and describe its drawback and then we will talk about completely new approach, that is also an exact constraint satisfaction approach but more efficient.

3.1 Naive Model

Here we will introduce a first constraint model for the protein prediction with side chain, and we will discuss the problems of this model. First of all to define our model we have to refer to the definition of the structure in the side chain model (2.2.1).The first condition guarantees the neighborhood condition between each side chain and its backbone .The second condition means that any two neighbors in the sequence should have backbones which are neighbors in the structure too (in our case the structure is restricted to the lattice, to simplify constraining). While the third condition means that any backbone or side chain of the amino acid should have a unique coordination in the structure (self-avoiding). Now to set up our model, we need to minimize the energy needed for the structure. The energy function simplification is given as:

$$Energy = -HHcontacts(seq, str) + HHchain(seq) \tag{3.1.1}$$

[22]

where $HHcontacts(seq, str) = |\{\{i, j\} : seq_i = seq_j = H \ \& \ str_i.sch, str_j.sch$ are neighbors $\}|$, and the HH-chain(seq) is the number of H followed by another H in the sequence $|\{i : seq_i = seq_{i+1} = H\}|$. A structure is called native if and only if it has minimal energy , and therefore it is our interest to minimize the energy. We can see that to minimize the energy we have to maximize the HHcontacts and minimize because the HHchain function depends only on the sequence and therefore it is constant for a given sequence and does not change if the structure of that sequence changed. Therefore our goal will be to predict a structure such that maximize the HHContacts. Let's try to give a constraint model such that guarantees maximizing the number of contacts in a cubic lattice (The constraint model is almost independent from the lattice shape, but the only difference is the definition of the neighboring constraint, since the neighbor vectors are changing according to the lattice shape). We suppose that sX,sY,sZ are the variables arrays representing the domains of side chains coordinates, and bX,bY,bZ those representing the domains of the backbones. Each element of the store is bounded between 0 and $2 * |sequence| - 1$, so denoting the sequence as seq, we have:

$$\begin{aligned} &\forall a \in sX_0, ..., sX_{|seq-1|}, sY_0, ..., sY_{|seq-1|}, sZ_0, ..., sZ_{|seq-1|}, \\ &\qquad bX_0, ..., bX_{|seq-1|}, bY_0, ..., bY_{|seq-1|}, bZ_0, ..., bZ_{|seq-1|} \\ &\qquad\qquad \Rightarrow a \in [0..2 * |seq| - 1] \end{aligned} \tag{3.1.2}$$

According to the definition of the protein structure in the side chain model we have to constraint each backbone in a neighboring constraint as denoted in the definition of the structure in this section above. And thus, we have the following constraint on the backbones:

$$\forall i < |seq| - 1 : |bX_i - bX_{i+1}| + |bY_i - bY_{i+1}| + |bZ_i - bZ_{i+1}| = 1 \tag{3.1.3}$$

Another constraint should be added which denotes the neighboring constraint between the backbones and side chains:

$$\forall i <= |seq| - 1 : |bX_i - sX_i| + |bY_i - sY_i| + |bZ_i - sZ_i| = 1 \tag{3.1.4}$$

Each side chain or backbone should be distinct, and therefore we have to propagate self-avoiding constraints:

$$\forall i \neq j : |sX_i - sX_j| + |sY_i - sY_j| + |sZ_i - sZ_j| \neq 0 \tag{3.1.5}$$

$$\forall i \neq j : |bX_i - bX_j| + |bY_i - bY_j| + |bZ_i - bZ_j| \neq 0 \tag{3.1.6}$$

$$\forall i \neq j : |bX_i - sX_j| + |bY_i - sY_j| + |bZ_i - sZ_j| \neq 0 \tag{3.1.7}$$

The above constraints are sufficient to guarantee that each monomer of side chain and backbone is distinct. The last step to come up with a complete model is to maximize the number of contacts that can be got with for H's side chains to minimize the energy function. To achieve this task will assign a reified boolean variable for each possible H side chain contact, and therefore we will need a reified array of boolean variables. Denoting this array as $Contacts_{i,j}$ we propagate:

$$Contacts_{i,j} \leftrightarrow |sX_i - sX_j| + |sY_i - sY_j| + |sZ_i - sZ_j| = 1 \tag{3.1.8}$$

Now our aim is to maximize the true values of the *Contacts* array, which means:

$$maximize(\sum_{i,j,seq[i]=seq[j]=H} contacts_{i,j}) \tag{3.1.9}$$

Maximizing the sum of contacts should be done through branch and bound approach, which is a general approach for finding the optimal solutions of various optimization problems (especially in constraint programming, and linear programming) during the branching. Each solution is constrained to be at least as good as the previous solution. The branch and bound approach was first proposed by A.H.Land and A.G.Doig (1960) [12]. This model is theoretically sufficient to optimize the energy problem but yet not efficient. There are many reasons of the inefficiency produced by this model. For example the reified propagator produced by constraint 3.1.8 is very heavy since it needs obviously to propagate $\frac{n*(n-1)}{2}$ reified propagators which is both time and memory consuming. The constraints 3.1.5, 3.1.6, 3.1.7 need also to propagate $\frac{n*(n-1)}{2}$ propagators but they are not reified and also they can be really efficient if this model was modified so the monomers got indexed through a specific encoding and then we can apply the distinct propagator which is pretty efficient, as we will explain later when we talk about the efficient way of constructing the hydrophobic core. Other problem will be branch and bound strategy. The branch and bound strategy guarantees in each searching step to obtain the solutions that are at least as good as the last solution. However the branch and bound in this model lacks the ability to begin with a tide bound of the number of contacts. Thus, since we do not know yet the maximal bound of contacts, it is still inefficient to use the branch and bound alone, beginning only with the best number of contacts the search engine could find. In general the optimization problem of the protein shape prediction in HP Model both with side chain and without, showed to be an NP-hard problem in many studies [11],[3]. Of course it is possible to come up with a statistical approach for the protein prediction [14] (And even the mentioned approach is polynomial), however we are not interested very much in a possible approximated solution, because as shown in [27] the degeneracy problem appears for the exact solutions in the backbone protein model and we believe it is also the case in this side chain backbone model,and therefore finding a sub-optimal solution is not interesting among large amount of exact solutions. Especially knowing as we will show in the result chater, that the degeneracy for the side chain model is much more than the "one amino acid- one monomer" representation, because of the freedom produced in placing the backbones and other factors. So we will continue to concentrate on building a constraint model, but we want this model to be realistic so we can get reasonable results. The general model that was proposed by [27]is believed to work too for the side chain-backbone model if a proper extension was made. Therefore we will introduce here briefly the extend model.

3.2 General View of the New Model

The general model of a full exact protein shape prediction for the side chain-backbone model is an extension to that without side chain (proposed by [27]), with the main difference in the last phase (Threading phase), as well as our definition of the Hydrophobic core should be changed a little to match only the core of the side chains of Hs. We will briefly describe the model, and we will talk in details about our work in building the hydrophobic core of the side chains, and threading the side chains and backbones to the core. First let's refer to the definition of the hydrophobic core for side chain model (definition 2.4.1). Now the general model scheme consists of two offline phases and one online phase. The new model is described by the following steps:

1. **Generating the Frame Sequences:** The first step is to generate the frame sequences for specific number of Hs and a bound number of contacts. The algorithm used is that suggested by Yue,Dill ,[28],[11] CHCC . The algorithm is previously also implemented and extended by Will [27] but we will give a short explanation of it because the output of this algorithm is needed in constructing the hydrophobic core. The idea of this algorithm and its extension is to generate all possible frame sequence sets so that we can build the hydrophobic core by constraining it to the generated frame sequences (Each layer should match at least one frame sequence). This is done through dynamic programming as we will briefly explain. First of all our input is the upper bound number of contacts as well as the number of points (in our case the number of points represents the number of dipoles mapped to the Hs side chains). The output of the CHCC (Yue, Dill approach) will be the sequence of layers where each layer is represented by its width and height as well as the number of points that it contains. Notice that then total number of contacts between the layers and inside each layer will be bounded by the given upper bound number of contacts. The contacts between the layers and inside each layer will be determined according to the landscape of Lattice. Since this algorithm is not part of this thesis -however is important to understand because its output is the input of our work, we will not go to the details of calculating the upper bound of contacts, and for more details you see it in [27]. For now we can write and describe the recursive equation for the frame sequences that is necessary to achieve the dynamic programming model for frame sequences generation.

$$B_C\left(n, n_1, a_1, b_1\right) = \begin{pmatrix} B_{LC}\left(n_1, a_1, b_1\right) \\ +B_{ILC}\left(n_1, a_1, b_1; n_2, a_2, b_2\right) \\ +B_C\left(n - n_1, n_2, a_2, b_2\right) \end{pmatrix} \tag{3.2.1}$$

where the function $B_C(n, n_1, a_1, b_1)$ is the upper bound of the contacts of a core with the size n, with the first layer contains n_1 number of points, and a_1, b_1 frame dimensions, $B_{LC}(n_1, a_1, b_1)$ is the upper bound of the contacts within the first layer, and $B_{ILC}(n_1, a_1, b_1; n_2, a_2, b_2)$ is the upper bound of the contacts between layer one 1 and its following layer, and the last function $B_C(n - n_1, n_2, a_2, b_2)$ is

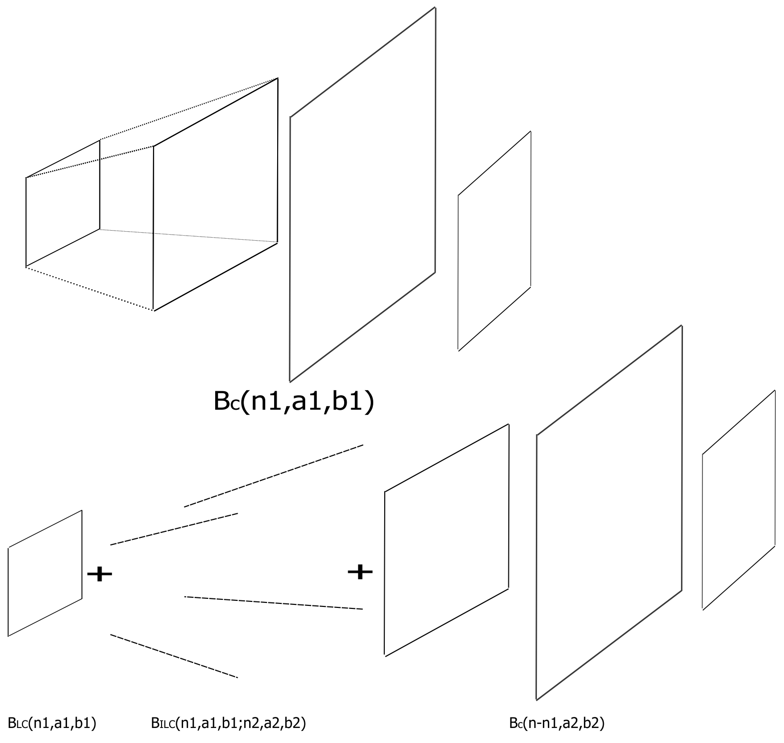


Figure 3.1: In this figure we show the recursion defined in the upper bound function on the contacts of a core, this recursion allows dynamic programming.

the upper bound of the contacts of the whole core excluding the first layer. We can easily recognize that the first two functions can be immediately calculated for given parameters while the third one is the recursive function on which the dynamic programming should allocate.We recognize too that for the second function $B_{ILC}(n_1, a_1, b_1; n_2, a_2, b_2)$ we have to find a method of bounding the contacts between too layers. It is not in the interest of this thesis to go through this, but for more details you can review the doctor thesis of Will [27] The figure 3.1 shows the recursion of calculating the bound contacts.

2. **Constructing the HCore:** After generating the frame sequences we have to build the landscape of the side chains of the H's within the lattice, and we will call the set of coordinates representing the hydrophobic side chains HCore

(Again the concept of the HCore here is different from its concept in the simple representation without side chain). Constructing the HCore uses the generated frame sequences as a strong constraint, and we will explain this in details in the next chapter.

3. **Threading the String Sequence to the Core of H's Side Chains:** The last step of the model is to thread the side chain of each hydrophobic amino acid to the core respecting the neighboring constrains that controls the distribution of the back bones of the hydrophobic amino acids outside the core as well as distributing the polar side chains and backbones outside the core respecting too their neighboring constraints inside the structure to match the neighboring relation in the sequence.

Since we already have a view about the concrete model we will talk in the next following two chapters in details about the second and third steps of the model.

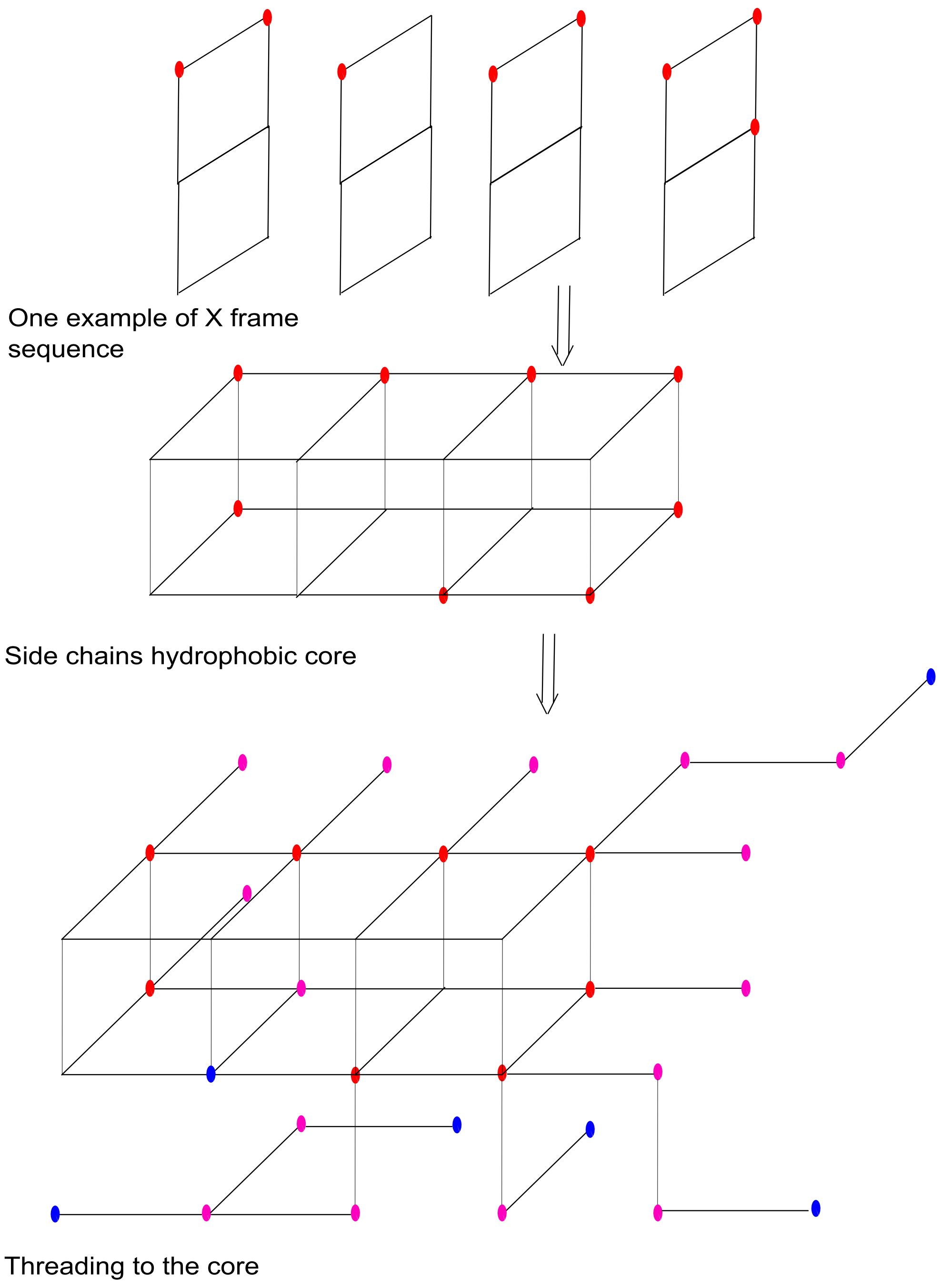


Figure 3.2: In this figure we see the three stages of the extended side chain-backbone model. We can notice that the three phases are almost identical to the model of simple representation (without side chain), knowing that in the last phase of threading, we should thread both the side chains and the backbones of the sequence to the side chain hydrophobic core. For the threading phase in the figure we can see, the polar atoms in blue, and the hydrophobic as red, and both backbones are pink.

Chapter 4

Constraint Models for Constructing The Hydrophobic Core

In this chapter we will explain two constraint models to build the hydrophobic core (the monomers representing the Hs side chains) in a way that maximizes the number of contacts between those side chains. We have the advantage now of building those cores before hand, without the necessity of reading any sequence. Our aim is building all possible cores that match a specific set of frame sequences with some number of contacts. Of course when we begin with the optimal number of contacts we are generating the optimal Hs side chains cores, but it is not enough to generate the optimal set of cores because it might not be enough in the threading part, and thus we will need many sub optimal level of cores. The level of sub optimality needed is sequence dependent and we will give some result about this in the last chapter. We will begin now with our definition to the two models discussing formally the applied constrained.

4.1 First Constraining Model

In this first model we begin with defining the variables of our constraint model. We will have three sets of Integers variables one for the X of each point and one for Y and one for Z as the following:

$$X = \{X_0, ..., X_{pointsNum-1}\}$$

$$Y = \{Y_0, ..., Y_{pointsNum-1}\}$$

$$Z = \{Z_0, ..., Z_{pointsNum-1}\}$$

where pointsNum indicates the number of the points. Now we have to define also our stores that should bound each variable as following:

$$store(X) = \{0, ..., maxLayer\}$$

$$store(Y) = \{0, ..., maxLayer\}$$

$$store(Z) = \{0, ..., maxLayer\}$$

where maxLayer is the maximum possible layer according to the sequences set. One of the remarks in the main characteristic of this model is that each variable is represented as integer variable, and thus each H side chain that occupies some point in the core will be represented by three variables (one for the X, one for the Y and one for the Z as shown before). Now we can begin defining our constrains that shape this model.

1. **X Constraint on one layer sequence:**

   For a given Layer sequence j;

   $$FrameSeq_j = \{N_{j0}, ..., N_{jL}\}$$
   $$N_{jr} : 0 \leq r \leq L$$
   $$store\,(X_i) = \{r\} : step\,(r) \leq i \leq step\,(r) + N_{jr-1}$$

where:

$$step\,(r) = \left\{ \begin{array}{r} 0 : r = 0 \\ step\,(r-1) + N_{jr-1} : r > 0 \end{array} \right\}$$

This constrain guarantees a refied constraint will be propagated to check if the dipoles in the X layers match a specific layer sequence.

2. **Y Constraint on one layer sequence:**

   For a given Layer sequence j; $FrameSeq_j$={ N_{j0},...,N_{jL} } .

   In general the number of Y_i where $Y_i = r$ should be equal to N_{jr}, where $0 \leq r \leq L$ and $0 \leq i \leq pointsNum$. To implement this in the gecode2.0 we have to formalize the problem in the following way: We reify the equation $Y_i = r$ by a Boolean variable let's call it B_r. So:

   $$Y_i = r \Leftrightarrow B_r\,(0 \leq i < pointsNum, 0 \leq r \leq L)$$

   $$\sum_r B_r = N_{jr}$$

   This also propagates a reified constraint to check the matching between the monomers number in the Y Layers and a specific element of the layers sequence set.

3. **Z Constraint on one layer sequence:**

   In exactly the same way of constraining Y$_i$.

4. **Global constraint:** So far each constraint is propagated as a reified propagator to see whether a specific Layer sequence was met. What we globally need is that each dimension's layers should meet at least one sequence. Let's denote the Sequences Set as FrameSeqSet where:

$$FrameSeqSet = \{Frameseq_0, ..., Frameseq_K\}$$
$$Where$$
$$FrameSeq_j = \{N_{j0}, ..., N_{jL}\}$$
$$0 \leq j \leq K \; and \; N_{jr} \in N \;\; 0 \leq r \leq L$$

Now let's consider the following reified functions:

$$constrainXOn(Frameseq_j) = 1 \Leftrightarrow$$
$$XConstraint \; on \; one \; Frame \; sequence \; is \; met \; for \; Frameseq_j$$

$$constrainYOn(Frameseq_j) = 1 \Leftrightarrow$$
$$YConstraint \; on \; one \; Frame \; sequence \; is \; met \; for \; Frameseq_j$$

$$constrainZOn(Frameseq_j) = 1 \Leftrightarrow Z$$
$$Constraint \; on \; one \; Frame \; sequence \; is \; met \; for \; Frameseq_j$$

Let's have now three arrays of reifying Boolean variables one for X, one for Y and one for Z:

B_X, B_Y, B_Z

We apply now:

for each $Frameseq_j$ in FrameSeqSet

$$B_{Xj} = constrainXOn(Frameseq_j)$$
$$B_{Yj} = constrainYOn(Frameseq_j)$$
$$B_{Zj} = constrainZOn(Frameseq_j)$$
$$\sum_j B_{Xj} = 1$$
$$\sum_j B_{Yj} = 1$$
$$\sum_j B_{Zj} = 1$$
$$0 \leq j \leq K$$

We notice here the disjunction propagators which unfortunately do not have an efficient propagator. However we will explain later a way of compromising such that we do not apply the disjunction propagators to all constraints but at the same time we do not fully loose the information exchanged through the constraints that are propagated disjunctively (Be cause if we do not want the disjunction propagator we will need to separately propagate the constraints and we will loose some useful exchanged information between them, as well as the problem of multiplexing the search space).

5. **Ordering constraint:** This model (just like the naive model) lacks the property of being ordered. Therefore we will get many symmetrical solutions for a single solution but differently ordered. Thus, In order to break one kind of symmetry which is preventing giving the same solution but in different order we have to apply the ordering constraint which is:

$$X_i \leq X_{i+1}$$
$$X_i = X_{i+1} \Rightarrow Y_i \leq Y_{i+1}$$
$$Y_i = Y_{i+1} \Rightarrow Z_i < Z_{i+1}$$
$$0 \leq i < pointsNum - 1$$

Where This ordering constraint will prevent an identical solution to be replicated, and will cover the problem of "non-indexed entries" introduced by the model

6. **Contacts constraint:** So far the introduced constraints are sufficient (yet not necessary efficient) to satisfy the layers sequences set without eliminating geometrical symmetry. The only remaining constraint to satisfy is the boundary of contacts number. Let's consider the minimum number of contacts minContact. Since we already know the bound of the contacts we will add other constraint to make sure that this bound were met. In our model the nave way of doing so is to apply the following constraints:

$$|X_i - X_j| + |Y_i - Y_j| + |Z_i - Z_j| = 1 \Leftrightarrow B_i$$
$$i < j$$
$$\sum B_i = minContact$$
$$0 \leq i < pointsNum$$

But this constraint is very heavy, especially that summing constraint of two variables in the gecode is not yet available, but we will explain later some ways of making this sum in this model more efficient.

7. **More efficient ways of Contacts constraint:**

Definition 4.1.1. Let's divide the contacts to three kinds X contact, Y contact and Z contact with the following definition: Since now all the points are ordered (according to constraint number 5) so let's have the set of monomers P=$\{P_0, ..., P_{pointsNum-1}\}$
With the totally order relation
$i < j \Rightarrow P_i \prec P_j$
We say that two monomers P$_i$ and P$_j$ $j < i$ have an X contact if and only if
$X_i = X_j + 1 \& Y_i = Y_j \& Z_i = Z_j$
The same to have Y contact:
$X_i = X_j \& Y_i = Y_j + 1 \& Z_i = Z_j$
And Z contact:
$X_i = X_j \& Y_i = Y_j \& Z_i = Z_j + 1$

Now instead of constraining all pairwise P_i, P_j (the number of the constraint in this situation by this model $\frac{pointsNum*(pointsNum-1)}{2}$) we only propagate the following constraints;

for each Z contact we can only propagate:

$$(X_{i+1} - X_i) = 0\&(Y_{i+1} - Y_i) = 0\&(Z_{i+1} - Z_i) = 1 \Rightarrow A(BooleanVariable)$$

Because Z contact can only be contained in consecutive points.

For each Y contact we can only propagate:

$$(X_j - X_i) = 0\&(Y_j - Y_i) = 1\&(Z_j - Z_i) = 0 \Rightarrow B(BooleanVariable)$$

for j>i and j<L where L is the number of layers for the current propagation of the current X Layer or we can bound the L with the maximum number of layers for simplicity.

Because Y contact can only be contained between the current point and the next L points.

For each X contact we can propagate:

$$(X_j - X_i) = 1\&(Y_j - Y_i) = 0\&(Z_j - Z_i) = 0 \Rightarrow C(BooleanVariable)$$

for j>i and $j < L^2$ where L is the number of layers for the current propagation of the current X Layer or we can bound the $maxLayer^2$ with maxLayer is the maximum number of layers for simplicity.

Because X contact can only be contained between the current point and the next $maxLayer^2$ points (this case is not very nice).

And as before we have to sum over all Boolean variables and assign it to the bound of contacts number.

We can recognize that even with this better approach the number of propagator and complexity could still be expensive for calculating the X contacts, and the problem here that we can not know where is the next ordered point which has a larger X, so we have to check all the points between our current point and next

$maxLayer^2$ points so if the maximum possible layer is big enough this could be really expensive. Even though we could do better than this in this model and we can discuss that later when we talk about constraint number 9.

8. **Implied strong range constraint:**

   We can recognize that this model lacks the power of some kind of constraint. To get an example for this, let's look at the following case:

   We have only one Sequence Layer with the following sequence:

   9 9 9

   This can be only be met if the cubic lattice with points beginning from 0,0,0 and end with 2,2,2 were completely part of the Hydrophobic core. By the current model this will not be known till a good amount of propagation.

   But we can add a range propagation so that there will be no need for any propagation to discover that this is the only solution. To formalize this propagation let's consider we are filling one X layer of L1 monomers (the current number of current X in the current sequence is L1), so in this case we will have L1 points with the same x. Assume that we are in point number i in those points (numbered as 0 to L-1):

   We can prove that each point in this formulation has a specific range as following:

   $$Range(P_i) = [P_i, P_{maxLayer*maxLayer-(L1-i)}]$$

   and now since we calculated the range of indexes now we can calculate the range of the X,Y,Z of this point. Hence i is not the global index but rather as we said before the local index in the current X Layer. Now the range of the X,Y,Z of the point in an X Layer K is: $Range(P_i) = [P_{i1}, P_{i2)}]$ where

   $P_{i1} = (K, i \ div \ maxLayer, i \ \%maxLayer)$

   $P_{i2} = \big(K, \big[maxLayer^2 - (L1 - i)\big] \ div \ maxLayer, \big[maxLayer^2 - (L1 - i)\big] \ \%maxLayer\big)$

9. **Better number of contacts constraint:**

   From the previous we can improve our contacts constraints. We will use the same definition and formulation that we used in describing constraint number 7. Now we will add the following formulations:

Let's consider we are filling a specific X layer K1 that has L1 as the current sequence number with the next Layer K2 has L2 as the sequence number. Let's consider an arbitrary point in P_i in K1 such that $0 \leq i < L1$ and point P_j in K2 such that $0 \leq j < L2$; then Let's formalize the following:

Let's P_{i1}, P_{i2} be two monomers. We say that RangeIndx $([P_{i1}, P_{i2}]) = [i1, i2]$.

Now to have X contact between P$_i$ and P$_j$ the following condition should be met:

$$\begin{aligned} & RangeIndx\left(Range\left(P_i\right)\right) \cap RangeIndx\left(Range\left(P_j\right)\right) \neq \oslash \\ & \Rightarrow \\ & \left[i, maxLayer^2 - (L1 - i)\right] \cap \left[j, maxLayer^2 - (L2 - j)\right] \neq \oslash \\ & \Rightarrow \end{aligned}$$

We have one of those three conditions to be met:

- i=j
- $i < j \Rightarrow j \leq maxLayer^2 - (L1 - i)$
- $j < i \Rightarrow i \leq maxLayer^2 - (L2 - j)$

If none of those conditions were met we do not have to propagate any constraint for the X contacts. And in this model this condition will really reduce to the number of the contacts constraints.

4.2 Second Constraining Model (Boolean Model)

In this model we get a cube that contains $|X| * |Y| * |Z|$ monomers, were $|X|, |Y|$ and $|Z|$ are the number of layers X,Y and Z consecutively. We will have an array of Boolean variables that assign the value 1 for the point if it belongs to the HCore and 0 if it does not.

$$Monomers = B_0, ..., B_{maxLayer^3}$$

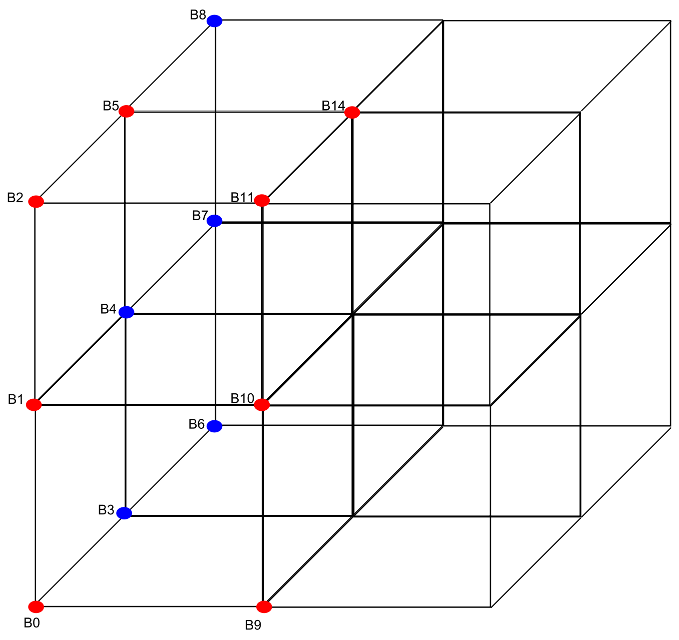


Figure 4.1: In this figure we see how can we encode our boolean variables. The red points belongs to the Core while the blue ones do not. Not all the points are numbered but an example of the boolean variables assigned to the points are shown.

where maxLayer is the maximum value for the layer. The indexes can be mapped to each point as following:

An arbitrary point $P_i\,(X_i, Y_i, Z_i)$ is mapped to B_{index} where:

$$index = Z_i + Y_i * maxLayer + X_i * (maxLayer)^2.$$

This model is similar to the model for constructing the hydrophobic core done and proposed by [27].

Now we define the following reification:

$B_{index} = 1 \Rightarrow P_{index}$belongs to the HCore. An example of encoding our variable is shown in figure 4.1.

Now we define the constraints:

1. **X Constraint on one layer sequence:**

Obviously in this model the sum of the Boolean variables that has a specific X value should be equal to the current number in layer sequence.

For a given Layer sequence j;

$$Frameseq_j = \{N_{j0}, ..., N_{jL}\}$$

For all N_{jr}: $0 \leq r \leq L$

$\sum B_i = N_{jr}$ where $r * L^2 \leq i < (r+1) * L^2$

2. **Y Constraint on one layer sequence:**

   The same idea for the Y the sum of the Boolean variables that has a specific Y value should be equal to the current number in layer sequence.

   For a given Layer sequence j; $Frameseq_j = \{N_{j0}, ..., N_{jL}\}$

   For all $N_{jr} : 0 \leq r \leq L$

   For all X: $0 \leq X < maxLayer$(where maxLayer is the maximum possible layer)

   we add the constraint:

   $\sum B_i = N_{jr}$ where $x * L^2 + r * L \leq i < x * L^2 + (r+1) * L$

3. **Z Constraint on one layer sequence:**

   The same idea for the Y the sum of the Boolean variables that has a specific Y value should be equal to the current number in layer sequence.

   For a given Layer sequence j; $Frameseq_j = \{N_{j0}, ..., N_{jL}\}$

   For all $N_{jr} : 0 \leq r \leq L$

   For all X: $0 \leq X < maxLayer$(where maxLayer is the maximum possible layer)

   For all Y: $0 \leq X < maxLayer$(where maxLayer is the maximum possible layer)

   We add the constraint:

   $\sum B_i = N_{jr}$ where $x * L^2 + y * L + r \leq i < x * L^2 + (y) * L + r + 1$

4. **Global constraint:**

   Now we have to make sure that the Boolean variables array constraints should meet at least one of the sequences list :

Let's denote the Sequences Set as FrameSeqSet where: Let's denote the Let's denote the Sequences Set as FrameSeqSet where:

$FrameSeqSet = \{Frameseq_0, ..., Frameseq_K\}$

Where *FrameSeq*$_j$={ N_{j0},...,N_{jL} } $0 \le j \le K$ *and* $N_{jr} \in N$ $0 \le r \le L$

Now let's consider the following reified functions:

constrainXOn$(Frameseq_j) = 1 \Leftrightarrow$ X Constraint on one Frame sequence was met for Frameseq$_j$

constrainYOn$(Frameseq_j) = 1 \Leftrightarrow$ Y Constraint on one Frame sequence was met for Frameseq$_j$

constrainZOn$(Frameseq_j) = 1 \Leftrightarrow$ Z Constraint on one Frame sequence was met for Frameseq$_j$

Let's have now three arrays of reifying Boolean variables one for X, one for Y and one for Z:

B_X, B_Y, B_Z

We apply now:

for each $Frameseq_j$ in FrameSeqSet

$$B_{Xj} = constrainXOn(Frameseq_j)$$
$$B_{Yj} = constrainYOn(Frameseq_j)$$
$$B_{Zj} = constrainZOn(Frameseq_j)$$
$$\sum_j B_{Xj} = 1$$
$$\sum_j B_{Yj} = 1$$
$$\sum_j B_{Zj} = 10 \le j \le K$$

We have to denote here that in all the previous constraining we have to fill the Boolean representing the monomers in the Layers that is not reachable or out of boundary in 0s instead of having a variable for the HCore boundary.

5. **Constraining the contacts:**

   Now constraining the contacts is much more efficient than the previous model. Let's consider the contacts bound contacts_bound.

   Then we can know the possible contacts for all monomers by summing the neighboring points Boolean values. So if we have to sum:

   $B_{i+1} + B_{i+maxLayer} + B_{i+maxLayer^2}$ $(0 \leq i < maxLayer^3)$
   Obviously we see that each point will maximally has one contact constraint which is a lot cheaper. Now our model is complete and will perform better (or as good as -in some cases-) the previous model. However we have to clarify some points. The disjunction propagator will be used for the sequences with specific dimensions. So for example the core handler will collect all the sequence number with the dimensions boundary 3,4,5 and apply the disjunction between them, and also for 4,5,5 and so on. And so we compromise between applying the disjunction propagator and applying the model for each 3 layers sequences as separated problem. This way will also break some symmetries, because we consider for example dX=3,dY=4,dZ=5 (where dX is the number of X Layers, dY then number of Y Layers, and dZ the number of Z Layers. In other word since the layers of each dimension are consecutive dX,dY,dZ are the boundaries of X,Y,Z dimensions respectively) but we do not consider dX=4,dY=3,dZ=5, and therefore we break many symmetries. And according to that the endowing system for each index of a boolean variable representing a monomer B_{index} will be:
   $index = Z_i + Y_i * dZ + X_i * dZ * dY$. More formally (and in a programming language formal descirption) we will divide the dX,dY,dZ according to the following:
   $for(dX = minLayer; dX <= maxLayer; dX + +)$
   $for(dY = dX; dY <= maxLayer; dY + +)$
   $for(dZ = dY; dZ <= maxLaer; dZ + +)$
   In the next section we explain the way of breaking the remaining geometrical symmetry.

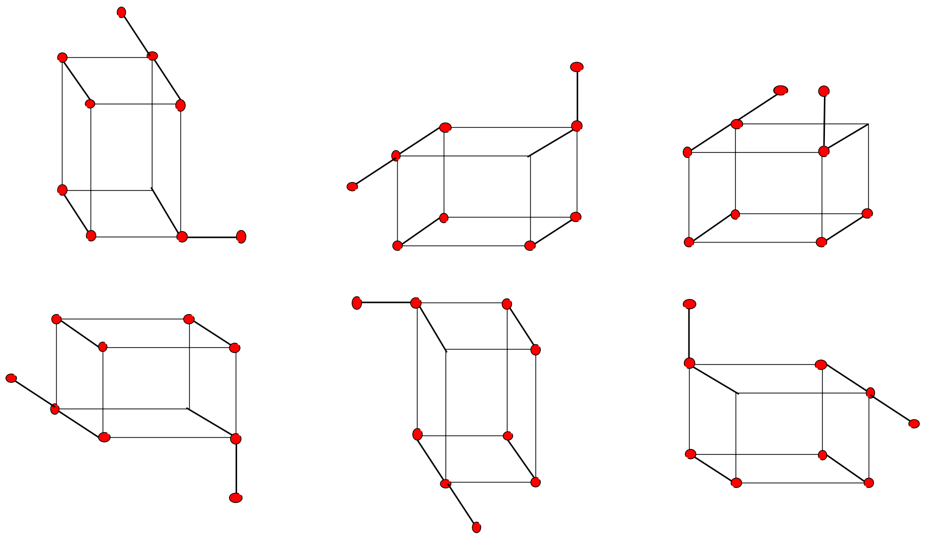


Figure 4.2: Here we have a hydrophobic core of 9 monomers and 10 contacts. All those solutions for the hydrophobic core are symmetrical and can be represented by one of them.

4.3 Breaking the Symmetries

Breaking symmetry is very important in order to avoid getting exponential solutions number in term of the identical solutions number. An example on how symmetrical solutions can be look like figure 4.2.

We have first to formally define symmetrical solutions. First let's denote a solution of the hydrophobic core as a set $P = \{p_1 = (X_1, Y_1, Z_1), ..., p_n = (X_n, Y_n, Z_n)\}$, and to let's denote p_i.k where $0 < i \leq n$ and $0 < k \leq 3$ as the coordinate number k in the 3-topple representing p_i, for example $p_i.1 = X_i$. Now:

Definition 4.3.1. We say that two non-identical solutions of the hydrophobic core,represented by two sets of points:
$P1 = \{(X_1, Y_1, Z_1), ..., (X_n, Y_n, Z_n)\}$
$P2 = \{(\grave{X}_1, \grave{Y}_1, \grave{Z}_1), ..., (\grave{X}_n, \grave{Y}_n, \grave{Z}_n)\}$

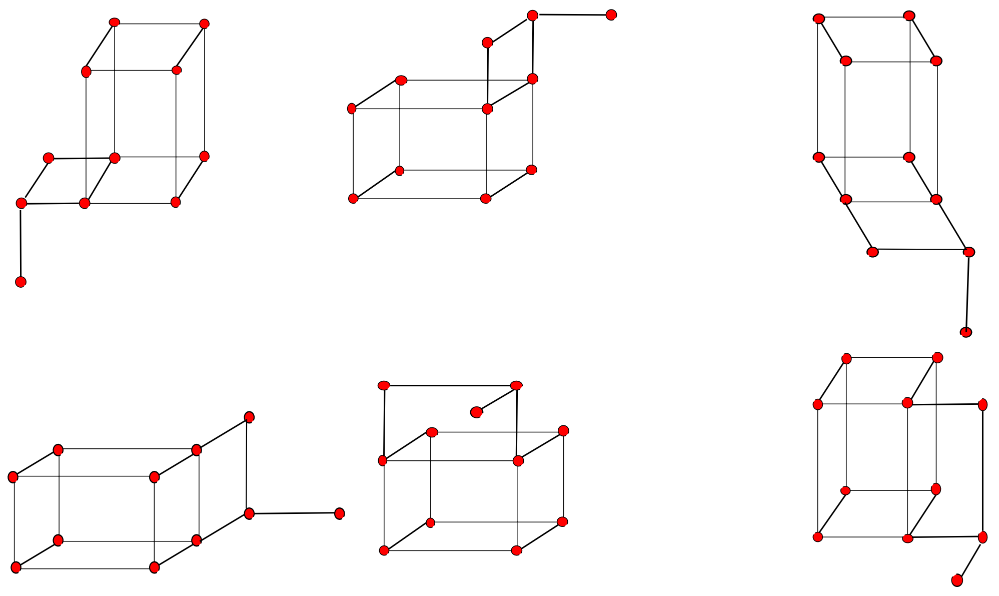


Figure 4.3: Here we have cores with 11 monomers with 16 contacts. And those cores are all the possible solutions that are symmetrical by rotation.

are symmetrical by "rotation only" if an onlly if:
$\forall p_1 \in P_1$ exists unique $p_2 \in P_2$, such that $\forall k : 0 < k \leq 3$ exists unique different i, where $0 < i \leq 3$ such that $p_1.k = p_2.i$.

This means that if we rotate a solution on the X, Y or Z axis we will get a symmetric solution. Note that the previous definition allows composing several rotation in case those rotation does not return back to the original solution. An example of the symmetrical rotation solutions is shown in figure 4.3.

Now considering the boundaries of the solution P_1 as dX, dY, dZ resp. for X,Y,Z dimensions, and $\widehat{dX}, \widehat{dY}, \widehat{dZ}$ the boundaries of P_2 resp. for X,Y,Z dimensions too, let's denote $d\xi_i$ where $0 < i <= 3$ as the boundary of dimension number i (1,2,3 for x,y,z respectively) in the P_1 solutions, and in the same way $\widehat{d\xi_i}$ represents the boundary of dimension number i in P_2 .

Definition 4.3.2. We say that two non-identical solutions of the hydrophobic core, represented by two sets of points:

$$P1 = \{(X_1, Y_1, Z_1), ..., (X_n, Y_n, Z_n)\}$$

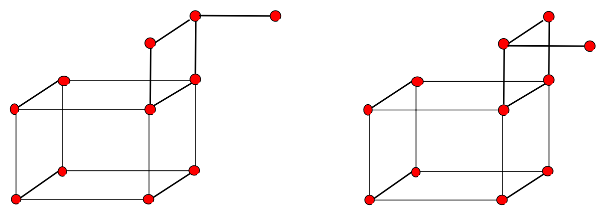


Figure 4.4: Here we have two cores with 11 monomers with 16 contacts. And those two cores are symmetrical by reflection.

$P2 = \{(\grave{X}_1, \grave{Y}_1, \grave{Z}_1), ..., (\grave{X}_n, \grave{Y}_n, \grave{Z}_n)\}$
are symmetical by "reflection only" if and only if at least for one dimension with number k $0 < k \leq 3$, $\forall p_i \in P_1$ exists unique $p_j \in P_2$ such that $p_i.k = \widehat{d\xi_k} - p_j$, or the symmetrical way around, which means:
at least for one dimension with number k $0 < k \leq 3$, $\forall p_i \in P_2$ exists unique $p_j \in P_1$ such that $p_i.k = d\xi_k - p_j$.

This definition will include all the solutions produced by reflection on the X,Y or Z axis. This includes several reflections, such that we do not get a different solution. An example of the reflectional symmetry can be seen in figure 4.4.

Finally we can formally define when two solutions of the hydrophobic core are symmetrical.

Definition 4.3.3. We say that two non-identical solutions of the hydropic core are symmetric if they were symmetrical by "rotation only" or symmetrical by "reflection only" or if one of them is symmetrical by "rotation only" to a symmetrical solution by "reflection only" to the other or if one of them is symmetrical by "reflection only" to a symmetrical solution by "rotation only" to the other.

Actually the easy way of expressing this is by rotating a specific solution and reflecting it and both reflecting and rotating this solution we should get all kind of symmetries. An example for this is in figure 4.5.

Actually by this we can see that we will have 48 number of symmetries, and the reason is that we have 3! = 6 number of symmetries for rotation alone including the

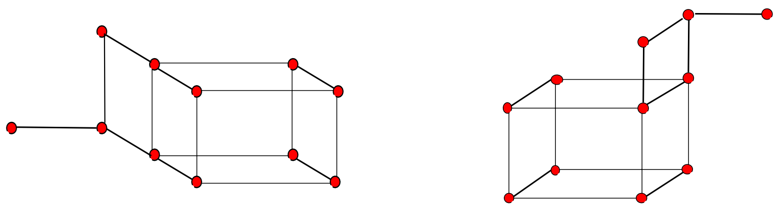


Figure 4.5: Here we have two symmetric cores by rotating then reflecting each other.

original solution,and for each case we can have $2^3 = 8$ symmetrical solution produced by reflections in 3 dimensions (one for reflection with rotation , and one for rotation without reflection, of course the case of reflection alone is included since the 6 number of symmetries produced by rotation alone includes the original solution).Thus, the number of all symmetries are $3! * 2^3 = 48$. Now we want to find a way that eliminates the symmetrical solutions with preserving all the non-symmetrical solutions, in order not to grow exponentially with our number of solutions. In the constraint programming generally breaking the symmetry is still an open problem. However there are usually many ways of breaking the symmetry, those ways can be introduce either in the constraint model through introducing other constraints to the store, or during the search to eliminate any new solution that is symmetric to a previously founded solution. However it is important in the first way, while introducing new constraint to introduce constraints that does not eliminate any different solution from the store.In our case most probably that it is not possible to introduce any constraint without possibly eliminating different solutions. Therefore now we have to introduce a way to break the symmetry during the search time. The main idea of this, is to add a new constraint during the search such that prevent introducing new solution that is symmetrical.

The general way of breaking the symmetry during the search proposed by Backofen and Will [4]can be seen in figure 4.6. This general view won't be complete till we mention that we need to recursively apply the new constraint to all right branches of the search. It is also important to know that this strategy is correct if and only if we are getting all solutions, otherwise we might ignore some solution during the search.

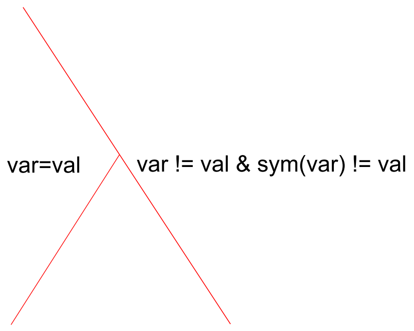


Figure 4.6: Here we see how we added a new constraint on the right branch of the search. This constraint will guarantee that the symmetrical variable should be unequal to the value assigned to the left side of branching. This will guarantee eliminating symmetrical solutions if applied recursively to all right branches.

Also The most important thing to mention here is that if we broke some symmetries (like in our case as explained in the previous section) we have to tell about those symmetry so we do not insert the new "breaking-symmetry" constraint, otherwise we will loose solutions. For each value actually we have 48 solutions but we already broke 6 of them in case $dX \neq dY$ & $dX \neq dZ$ & $dY \neq dZ$ and 3 of them in case two boundaries of two dimensions are equal and the third is different (for example $dX \neq dY$ & $dY = dZ$) and none of them are broken if $dX = dY = dZ$, because of the way we propagate as described in the previous section. Thus, in case some of the symmetries are broken we have to tell about those symmetries. We have a systematic way of numbering the 48 symmetries. We apply rotation for each reflection, and we begin reflection from z ending with x and we begin rotating around x ending around z, so we for each variable we get the following symmetrical variables (then we might transfer to the indexing version according to our encoding system) :

No Rotation:

x , y , z

x , y , dZ-z

x , dY-y , z

x , dY-y , dZ-z

dX-x , y , z

dX-x , y , dZ-z

dX-x , dY-y , z

dX-x , dY-y , dZ-z

X rotation:

x , z , y

x , y , dZ-y

x , dY-z , y

x , dY-z , dZ-y

dX-x , z , y

dX-x , z , dZ-y

dX-x , dY-z , y

dX-x , dY-z , dZ-y

and we continue numbering like this. This will automatically assign each symmetry for each variable to a number, and this allows us easily too to mark the broken symmetries, and thus very useful in the technical implementation.

Chapter 5

Threading protein sequences into cores in side chain - backbone model

The aim of this chapter is describing the last step of predicting the optimal structure of a side chain-backbone HP protein model. As we saw in the introduction, the amino acid general structure consists of the Amino group and Carboxyl group recall (figure 1.1). The sequence of amino acids is connected by peptide bonds. Each peptide bond connects an amino acid with the next amino acid, where the carboxyl group of the first amino acid reacts with the amino group of the second one (recall figure 1.2). We want now to model the structure of the protein considering each amino acid as two neighboring monomers one for the side chain and the other for the backbone instead of one monomer for each amino acid. In this model only the connection between two side chains of H amino acids is established through -1 energy state, for all other cases the energy state will be neutral (which means 0). Therefore in our model we will constrain threading the side chain of each hydrophobic amino acid to the core, everything else (polar double monomers, and hydrophobic backbones) should be outside the hydrophobic core. In this model we can here recognize that since each side chain must be connected to a neighboring backbone a hydrophobic core with maximally compacted H-contacts that fit to a thread of the one-monomer model might be not suitable of the double monomer model, so we might need to try several level of cores; denoting the level of a core by its number of contacts (level 0 has the maximal number of contacts), and the idea is: once we reach a level in which there exists a solution we do not look further in a higher level (lower compacted contacts). We will use both the Cubic lattice model and the Faced Centered Cubic lattice FCC as standard lattices to place the double monomers of each amino acid in the protein sequence. The constraining model that we will describe is almost independent from the lattice graph, as only the neighboring constrain differs when changing the graph as well as some extra implied constrains that can be added depending on the graph kind. In our model we will first explain our encoding for the constraint variables, then we will show the constrains.

5.1 Encoding the variables of the constraint model

Since we will need to apply several powerful constraint such as the self-avoiding it is much more efficient to compile each point denoted as $P_i\langle X_i, Y_i, Z_i\rangle$ to a single integer value denoted as $P_i = X + Y * LB + Z * LB^2$ (And of course weighing each coordinate could be done in other order like weighing Z by LB^0) Where LB is the length of the boundary of the lattice. The easiest way before threading the sequence is by choosing the boundary of the lattice as twice as the length of the amino acids sequence in the case of one monomer per amino acid model so that the hydrophobic core could be placed in the center of the lattice.Thus, we make sure that in the extreme cases all the monomers will fit in the lattice after threading as the figure 5.1 shows.

For our "Side chain - Backbone" structure we will need a bound lattice of size $[(seqLen + 2) * 2 \times (seqLen + 2) * 2 \times (seqLen + 2) * 2]$ in order to be able to fit the backbones as the figure 5.2.

Now for our model we will use four IntVarArray variables (each IntVar represents an indexed point p_i as we explained before), each of which has its own domain; therefore they are separated which will be more efficient when propagating. Let's define the following arrays:
$Hs = \{h_0, ..., h_{HCount-1}\}$, where HCount is the number of hydrophobic amino acids (the H's in the sequence) and $store(h_k) = hullLevel(hCore, 0)$ where $0 \leq k < Hcount$ and hCore is our current Hydrophobic core in which we are threading the amino acids sequence and the hullLevel is a function that we will define soon. The Hs IntVarArray is actually representing the monomers values of the hydrophobic amino acids "Side Chain".

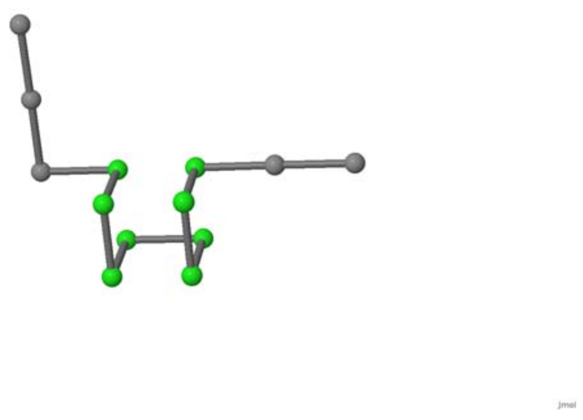


Figure 5.1: This is a valid solution for a simple sequence PPHHHHHHHHPPP. Here we can see the P's monomers to the sides of the HCore and therefore the HCore in this case is in the middle. In the figure the P's monomers are gray while the H's are grenn.

$BHs = \{bh_0, ..., bh_{HCount-1}\}$, where HCount is the number of hydrophobic amino acids (the H's in the sequence) and $store(bh_k) = hullLevel(hCore, 1)$ where $0 \leq k < Hcount$ and hCore as previously the current H-Core. Here the BHs array represents the monomers values of the hydrophobic amino acids "Backbones".
$Ps = \{p_0, ..., p_{PCount-1}\}$, where PCount is the number of hydrophilic amino acids (the P's in the sequence) and $store(p_k) = hullLevel(hCore, maxHull(hCore) + 2)$ where $0 \leq k < Pcount$ where we will also define maxHull function soon. Also obviously Ps array represents the monomers values of the hydrophilic amino acids "Side Chain".
The last variables array is: $BPs = \{bp_0, ..., bp_{PCount-1}\}$, exactly as before PCount is the number of hydrophilic amino acids (the P's in the sequence) and $store(bp_k) = hullLevel(hCore, maxHull(sequence) + 2)$ where $0 \leq k < Pcount$ and sequence is the string representing the amino acids sequence.
Now let's define the following concepts:

Definition 5.1.1. We define a set of monomers HL as the Hull level L of H-Core

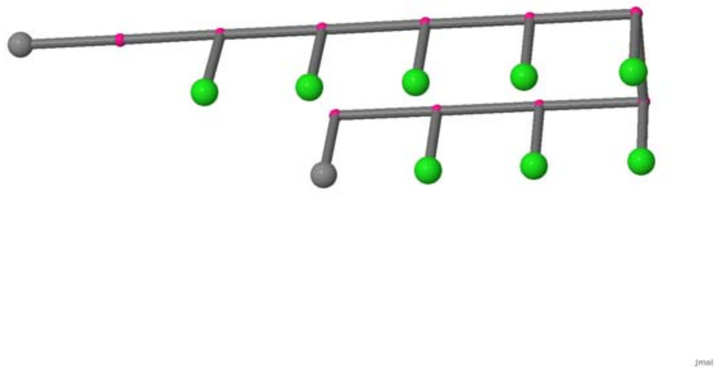


Figure 5.2: This is a valid solution for a side chain-backbone protein model for the sequence PHHHHHHHHP. We can notice that we need a lattice with dimensions of length (sequence length+2)*2 for each, otherwise we might eliminate solutions or at least break symmetries. In this figure the P's side chains are gray while the H's are green, and the backbones are pink.

$hCore$ if and only if HL contains all the monomers with at most L euclidian distance from the hydrophobic core excluding the H-Core monomers if $L > 0$, or the HL is the H-Core monomers set if $L = 0$

We denoted the Hull level function as hullLevel when we defined our variables arrays.

Definition 5.1.2. We define an integer M as the maximum hull of a sequence if and only if it is equal the maximal euclidian distance between an H in the sequence and other P.

We denoted the maximum hull function of a sequence as maxHull. For example in a sequence seq=HPPPHPPPPH, $maxHull(seq) = 2$
In a simple protein model where each amino acid represented by a monomer, the level of the domain set of each element of the Ps array can be bounded by $maxHull\,(sequence)$. However in the "Side chain-Backbone" Model the domain set of each element should be bounded by $maxHull\,(sequence) + 2$. The obvious reason for this, is that if a max hull value is M this means that the distance between at least one P backbone and H backbone is M which means that the distance between the side chain of those P and H is $M + 2$. Of course it is highly probable that the backbone of this P is connected with a backbone of another H with less distance within the protein structure, but we need to take the maximum possible distance to make sure we are not excluding solutions, unless we initialize the domain for each variable alone (which can also add more efficiency). An example for this is in figure 5.3.

5.2 Threading constraint model

Now we will describe constraining our variables. We will begin with our major constraints that are sufficient to come up with the optimal structure, then we can add some implied constraints and tricks that will increase the efficiency of our application. As we mentioned the constraints are almost lattice graph shape independent (in most cases only the neighboring constrain is graph shape dependent). The major constraints to apply are:

1. **In core constraint:** This constraint should guarantee that all side chains of the H amino acids should be in the hydrophobic core. Here we do not

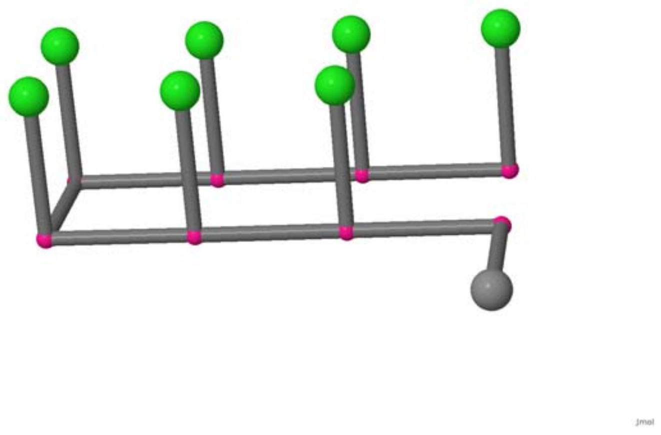


Figure 5.3: In this figure we have the simple sequence PHHHHHHH. We can see clearly according to our definition to the max hull level, that the max hull is 1 but though the maximum possible distance from the core in the side chain-backbone model is 1+2=3. This is because of the spaces occupied by the backbones. In this figure again the side chains of the P's are gray and for the H's are green, and the backbones are pink.

have to propagate any constrains since our definition of the variables domains guarantees to meet this constraint.

2. **Out core constraint:** In the same way this constraint should guarantee the fall of all other monomers (The backbone of H amino acids + the side chains and backbones of P amino acids) outside the core. Again there is no need to propagate any extra constraint for this since the definitions of our variables domain will guarantee this in an efficient way.

3. **Neighboring constraint:** This constraint should make sure that each backbone of an amino acid should be a neighbor in the protein structure to all the backbones of the amino acids that they are neighbors to that amino acid in the sequence. We will talk about this constraint more formally.

4. **All different constraint:** This constraint should make sure that each monomer is not different to any other monomer, which means that each backbone and side chain for each amino acid has its own Indexed point value. we will talk also about this constraint more formally later.

5.3 Neighboring constraint

Let's consider a normal HP protein sequence $seq = \{p_0, ..., p_n\}$ where $p_i = H$ or $p_i = P, 0 \leq i \leq n$. In a simple protein model without a side chain-backbone for each amino acid, we define the neighboring propagator as following: $i < n \Longrightarrow \overrightarrow{V}_{IndexedPoint(p_i)} - \overrightarrow{V}_{IndexedPoint(p_{i+1})} \in NeighorVectorsSet$ Where $IndexedPoint(p_i)$ is the relation that maps each amino acid monomer to the integer variable representing the indexed point. While the $NeighborVectorsSet$ is the set of all neighbor vectors of our lattice. For instance in the Cubic Lattice $NeighborVectorsSet = \{(1,0,0),(0,1,0),(0,0,1)\}$.
In the FCC $NeighborVectorsSet = \{(\pm1,0,0),(0,\pm1,0),(0,0,\pm1)\}$.
However in the side chain-backbone protein model that is a little different. In our case each backbone of an amino acid should be a neighbor to all backbone of structure neighboring amino acids. Also each backbone should be the neighbor of the side chain of the same amino acid. Formally recalling our encoding model of the variables we should propagate the following constraint:
$h_i \in Hs, bh_i \in BHs, 0 \leq i < HCount - 1 \Rightarrow (\vec{h_i} - \vec{bh_i}) \in NeighborVectorsSet$

and in the same way:
$p_i \in Ps, bh_i \in BPs, 0 \leq i < PCount - 1 \Rightarrow (\vec{p_i} - \vec{bp_i}) \in NeighborVectorsSet$
Of course the Hs,Ps,BHs,BPs are previously defined according to our encoding to the problem, and HCounts, PCounts are resp. the numbers of hydrophobic amino acids, and polar amino acids in the sequence. Those first two propagators are to make sure of the neighborhood between the backbones and the side chains. Also we have to propagate the following constraints:
$bh_i \in Hs, 0 \leq i < HCount - 1 \Rightarrow (\vec{bseq_i} - \vec{bseq_{i+1}}) \in NeighborVectorsSet$
where $bseq_i, bseq_{i+1}$ the backbone of the amino acid seq_i in the sequence. Those two propagators are to make sure that each backbone should be neighbor in the protein structure to its sequence neighbors.

5.4 All different constraint

We propagate this constraint to make sure that each indexed point in our variables array is distinct (self-avoid). The previous constraint will guarantee 2 walk-avoid (each point is diffident from its neighbor), and now we should define a global constraint that satisfies self avoidance; Formally:
$\forall x_i, x_j \in Hs \bigcup Ps \bigcup BHs \bigcup BPs, i \neq j \Rightarrow x_i \neq x_j$
The All different constraint propagator is implemented efficiently in the Gecode library as the distinct propagator. The distinct constraints is strong as it is hyper-arc consistence as well as it is efficient with the complexity of $n^{2.5}$ where n is the number of elements in the domain,(Régin, A filtering algorithm for constraints of difference in CSPs, AAAI 1994) [18].

5.5 Implied constraints and extra checks for better efficiency

Here we will explain some tricks and checks as well as other constraints to make our model more efficient.

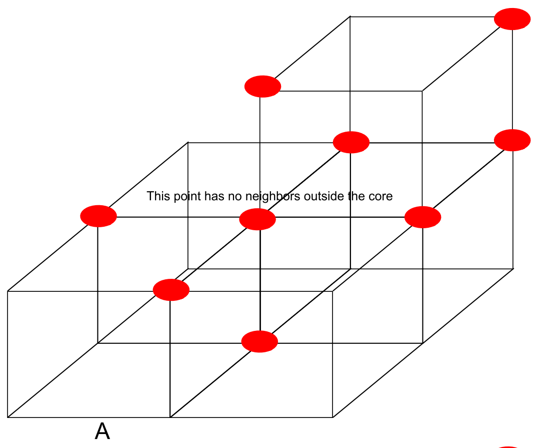
5.5.1 Parity Check in Case the Lattice is Cubic

In the case of the Cubic Lattice, Let's define the points into two classes:
$\mathbb{Z}^3_{even} = \{\vec{P}(x,y,z) | \vec{P} \in \mathbb{Z}^3, x+y+z \text{ is even } \}$
$\mathbb{Z}^3_{odd} = \{\vec{P}(x,y,z) | \vec{P} \in \mathbb{Z}^3, x+y+z \text{ is odd } \}$
Actually in case of the lattice Cubic we get the property that each two neighboring monomers should belong to different classes. Although, from this prospective the FCC will be a better representation from this prospective because we are not bounded by this property, the thing that gives us more freedom, but in case we want to make the prediction in case of the Cubic lattice, we can use this property to reduce the computation. Since we have to check each sequence against the current hydrophobic core, in case of Cubic lattice we can begin with checking the parity property before doing any propagation. If the parity property of the sequence does not match that of the Core, there will be no need to go in further propagation knowing that there will be no solution for a cubic lattice. Formally:
Let's consider $seqEvenPoints = \{i | seq_i = H \& \ i \text{ is even } \}$.
In the same way we define:
$seqOddPoints = \{i | seq_i = H \& \ i \text{ is odd } \}$
And we will define the same for the core:
$coreEvenPoints \subset \mathbb{Z}^3_{even} = \{\vec{P} \in \mathbb{Z}^3_{even} \wedge \vec{P} \in HCore\}$ where $HCore$ is the set of all core points. In the same way we define:
$coreOddPoints \subset \mathbb{Z}^3_{odd} = \{\vec{P} \in \mathbb{Z}^3_{odd} \wedge \vec{P} \in HCore\}$
Now we define :

Definition 5.5.1. We say that the hydrophobic core is compatible with the sequence if
$(|seqOddPoints| = |coreOddPoints|$ AND $|seqEvenPoints| = |coreEvenPoints|)$ OR $(|seqOddPoints| = |coreEvenPoints|$ AND $|seqEvenPoints| = |coreOddPoints|)$.
Now if the hydrophobic core was compatible with the sequence we constrain the sequence against the core otherwise we just skip this core, in case of the cubic lattice.

5.5.2 Filtering the Cores that Contains at Least One Point that has only Neighbors in the Core

Also h-cores filtering can be done to improve the efficiency. Each core should have all its points belonging to the surface (each point should be connected to at least one point that does not belong to the core). This condition is obviously resulted from the constraint that force the neighborhood relation between each side chain and its backbone. Since all the backbones do not belong to the core, we have to to guarantee that each point in the core is a surface point. Thus, we can already remove from the consideration each H-Core that does not match this property 5.4



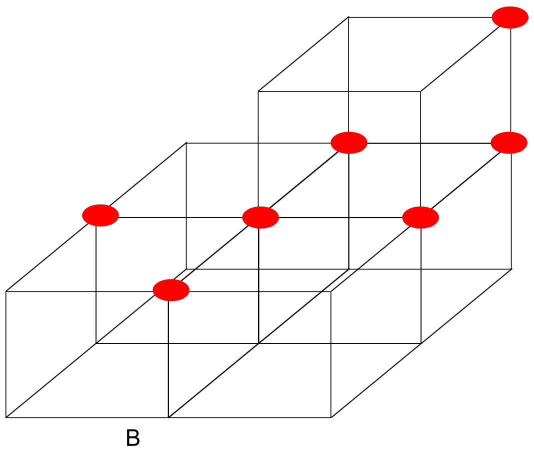


Figure 5.4: In this figure we can notice that the first core (A) needs to be filtered, while the other core (B) is accepted.

Chapter 6

Results and Application Area

In the previous chapters we gave a view of our theoretical model of protein structure prediction with side chain-backbone structure for each amino acid. In this chapter we will talk about the real life applications resulted from that model as well as the results obtained. Furthermore we will give a very strong argument that the CSP model is from many points of views and in this particular problem is much better than any statistical model that approach sub-optimality instead of the full-optimality. We are going to talk about the degeneracy problem and we will compare it to that of the simple protein model (without side chain).

6.1 Implementation

We implemented both constructing the hydrophobic core and the threading step for the side chain HP model. The implementation was done in c++ under Linux, as it is well known that most of the problems that need high efficiency are highly recommended to be implemented by c++. The HCore program was done as a stand alone program and uses the gecode library for constraint programming. The threading step was done in a way that allows easy integration to the cpsp library ([13]). It also uses both the cpsp library and biu library (also a library done by the bioinformatics team of the university of Freiburg) when necessary to avoid code duplications. Also a protein like sequences application was implemented to give statistics over random sequences. The Monte Carlo algorithm was re-implemented to use our Threading handler to generate successful protein like sequences. One of the advantage of the implementation that it allows a very similar command line interface that is provided by the biu and cpsp and that allows easy integration to the cpsp. Integration can be

Core size	Number of layers sequences	Time needed in second approach	Time needed in first approach
9	2	0.02s	0.03s
9	7	0.204	0.59s
9	27	1258.43s	Unknown
9	101	2-3 days	Unknown

Table 6.1: In this table we give some statistics over building the HCore. The consumed time is including streaming the output.

done for the side chain model threading library and the generated binaries (Threading, protein like, degeneracy statistics, side chain viewer, ...). Another advantage of the implementation, is that we implemented the viewer for side chain as a stand alone program, as well as an option that can be used to view directly during searching the solutions. In next sections we talk about the results obtained from this implementation.

6.2 Statistics over Constructing the Side Chains of the HCore

We will first give an over view of the performance got by the HCore model in the real life application. We will remind that in all constraint satisfaction programs we use the gecode c++ library, which was proven to achieve one of the best results in constraint programming. The statistics given here are for core of size 9 because we needed some extra level for the cores of size 9. And the core construction was done also for cores of much larger sizes like 36 but without many levels (2 levels of optimality for example). The second model always showed better performance and more practical from prospective of time. The time and hardness of solving the problem is not depending on the core size as much as on the number of possible frame sequences (for example it could be much more faster to get the result for the cores of size 36 than 9 when the cores level for 9 is higher (the compactness is lower, which means that the number of possible layers number is more), (table 6.2). In table 6.2 we give some comparison between the first and second models we explained in Chapter 4. We can see that the second model is always preforming better (matching the theoretical approach), and even we could not get the results in some cases for the second model. The statistics for big number of sequences (over 20) is not so clear, because if we eliminate some of those sequences the consumed time will change dramatically, and it is more dependent on those specific sequences than the number of sequences (for some sequences it seems very hard to decide if or if not exists a solution.)

In constructiong the hydrophobic core we are using till now one machine, but the big advantage of our model that it is very much support parallelization and we prepared a batch file that can distribute the tasks over many clusters but no experiment was done to get the results using parallel computing yet, although it is strongly believed that parallelizing will be very helpful in term of computational time. One of the nice thing here in the HCore is that it is done offline, and it is always demanded to enhance the computation time, but the demand is not so critical as in the cases of online computations. Constructing the HCore is a critical part of the study because if the database was suitably filled the next main step of exact prediction will with high success rate.

6.3 Results for Threading and Structure Prediction for the Protein with Side Chain

One of the important and most forward application area for the model is predicting the protein possible optimal structures. Since we already have a good database for HCores in both FCC and Cubic lattice we are going to give some results for predicting some protein structures given the sequences. The threading process proved to be pretty efficient and the success rate of finding an optimal solution using the current data base was relatively high in both Cubic lattice and FCC (especially in the experiments done among sequences of 30s or 40s amino acids). For human reading, we just want to shortly mention that the threading program allows the solutions to be printed both as x y z coordinates as well as moves (left L, right R, down D, up U ,forward F or backward B) where each move begin from a backbone towards its side chain then the next move towards the next backbone and so on . Thus each amino acid needs two moves, and the length of moves is as much as twice the length of protein sequence -1. You can get as much solutions as you want, and the model as we previously described

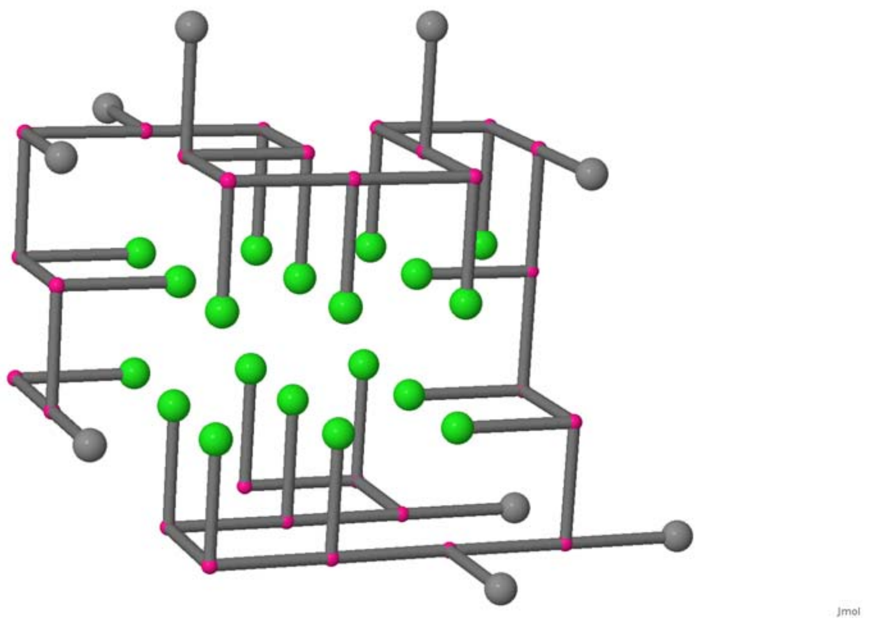


Figure 6.1: This is an example of the output figure of threading the sequence ”HPHHPPHHPHHHPHHPHHHPPHHHHPHH” into a cubic lattice in the side chain-backbone model. The H’s side chains are green, P’s side chains are gray and the backbones are pink.

guarantees the optimality of each solution regarding the energy function, and no sub-optimal solution will be introduced. We give in the following tables (6.3, 6.3), and figures (6.1, 6.2) some general results on some given sequences, with some predicted structures drawn in the lattice that was given in the query.

What we can recognize from the tables 6.3, 6.3 that sometimes it is more time consuming to get the solutions for the FCC and sometimes it is cheaper than the Cubic lattice. And the main reason for that depends on how difficult it is to get the solutions. Usually if the number of solutions is big in both cases (Cubic lattice and FCC), then the propagators could be cheaper in the Cubic lattice. But in the cases where the solutions are more difficult to find in Cubic lattice than the FCC then it is much more efficient to find solutions for FCC. An example for that is the sequence ”HPHHHPPHHHHHHPHHHHHHHHHHPH”; in this sequence for instance there are

Sequence	Time to get the first solution	Time to give maximum 10^6 solutions	Sequence length	Number of Hs	Number of reached solutions
HPHHHPPHHH HHHPHHHHH HHHHHPH	97.35s	141.86s	28	21	17280
HPHPPPPPHP HHPPPPPPPH PPHHHPHPPP PPHHPHPHP PHPPPHHH-PHP	0.76s	9.8s	50	19	10^6
PPPHHPHPPH PPHPPPHHHH PPPPHHPHHH PPPPPHHPP PHPHHHH-PHHP	579.89s	616.54s	50	23	10^6
PPPHHHHHPH HPHPPPPPHH PHPPHPPPPP HPPPPPPPP PPHHHPPPHH PHPPPPPH-HHH	1.71s	11.78s	60	23	10^6

Table 6.2: In this table, we see some sequences with different lengths with the time taken to thread them. In case the solutions are more than 1 million we stop searching for other solutions.

Sequence	Time to get the first solution	Time to give maximum 10^6 solutions	Sequence length	Number of Hs	Number of reached solutions
HPHHHPPHHH HHHPHHHHH HHHHHPH	0.76s	6.15s	28	21	10^6
PPPHPHHPHH PPPHPHPPPP PHPHHPPHPH HHHHHPHHP HPHPHPH	15.5s	24.9s	46	23	10^6
HPHPPPPPHP HHPPPPPPPH PPHHHPHPPP PPHHPHPHP PHPPPHHH-PHP	4.96s	14.12s	50	19	10^6
PPPHHPHPPH PPHPPPHHHH PPPPHHPHHH PPPPPHHPP PHPHHHH-PHHP	19.46s	29.4s	50	23	10^6
PPPHHHHHPH HPHPPPPPHH PHPPHPPPPP HPPPPPPPP PPHHHPPPHH PHPPPPPH-HHH	135.13s	146s	60	23	10^6

Table 6.3: In this table shows examples of threading sequences of different lengths in FCC lattice. The time given is for getting at most 1 million solutions. The time shown for the implementation is shown to be pretty reasonable You can also see the huge number of optimal solutions got for each sequence in the FCC.

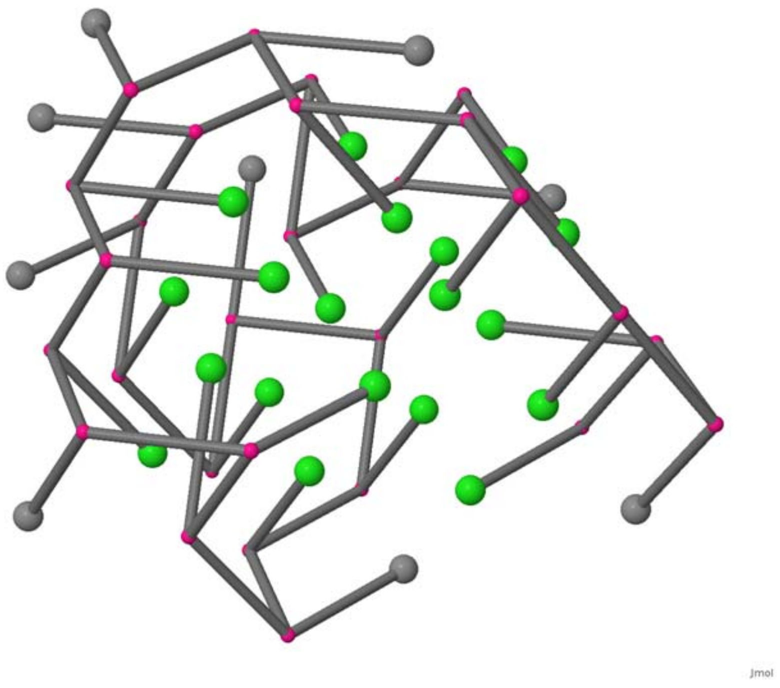


Figure 6.2: This is an example of the output figure of threading the sequence "HPHH-PPHHPHHHPHHPHHHPPHHHHPHH" into FCC lattice in the side chain-backbone model. The H's side chains are green, P's side chains are gray and the backbones are pink.

only few solutions in the cubic lattice (We see later why they are called few), and those solutions are hard to find and thus it is time consuming, unlike the case of FCC where we have large number of solutions (and we stopped only for one million, and if we explored all the search space, it could take more computational time for FCC). Other reason to make it harder sometimes to find solutions for the Cubic Lattice is that the HCore optimality is with less level than the FCC. But the question is, how many optimal solutions can we get for a given sequence, and is it practical to get all of them especially if we want to make another study on them? Was it a real advantage to do a constraint model instead of building a statistical model? We introduce the answer for those questions in the next section, when we study the degeneracy problem.

6.4 Degeneracy Statistics and Comparison with the Degeneracy in Backbone HP Model

Since we already know that the optimal energy structure in the side chain-backbone model is just like the simple model is not unique we are interested in having an overview of how much solutions a random sequence could have. According to [27] the number of solutions for each sequence (the degeneracy for each sequence) could be already large in the backbone model, and some sequences have more that million solutions. However in our case it turned out that the probability of having a very large degeneracy is much higher than in the simple monomer model (backbone model) and almost inevitable. Figure 6.3 gives a chart that is made up of almost 2250 sequences. It turned about that most of them have degeneracy in term of millions (more than 10^6). From 2243 sequences about 93% have degeneracy more than 1 million in case of Cubic lattice(could be hundreds of millions). A very minority of those sequences have degeneracy between 10^5 and 10^6 (only 15 sequences) which are only about 0.7%. There are about 5% of those sequences have not enough cores in the current database. We can look also to the logarithmic chart of the degeneracy according to the frequency and we can compare it to that in [27] and we can recognize that the degeneracy is much more in the side chain-backbone model (figure 6.4). The main reason of that big difference and huge amount of solutions for most of the sequences is the freedom resulted from placing the backbones as well as the Cores themselves are less optimal in many cases ad we filter all cores that have at least one point with has no neighbor

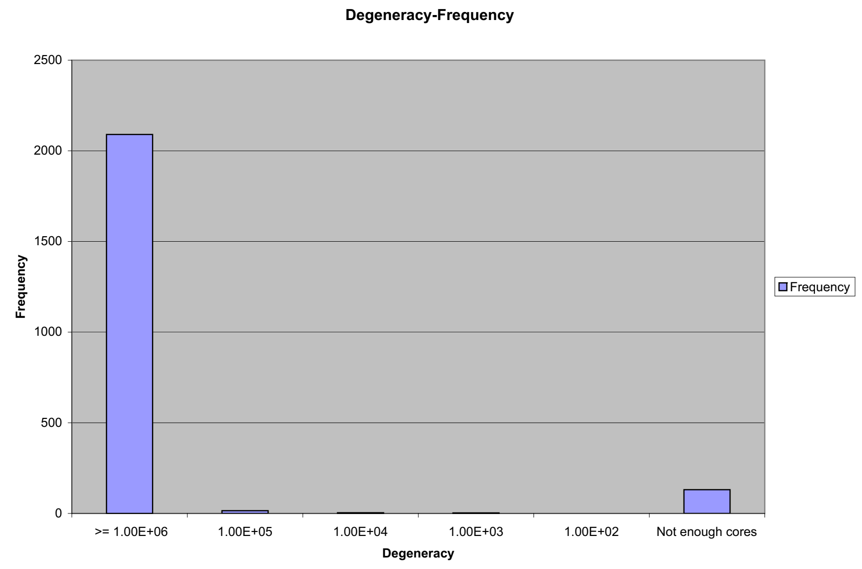


Figure 6.3: In this figure we have statistics about the degeneracy related the side chain-backbone model in a cubic lattice, and it shows that most of the sequences have over 1.0E+06 degeneracy and very few number of sequences have between 1.0E+05 degeneracy where almost 5% of the sequences do not have enough cores in the current database for calculations.

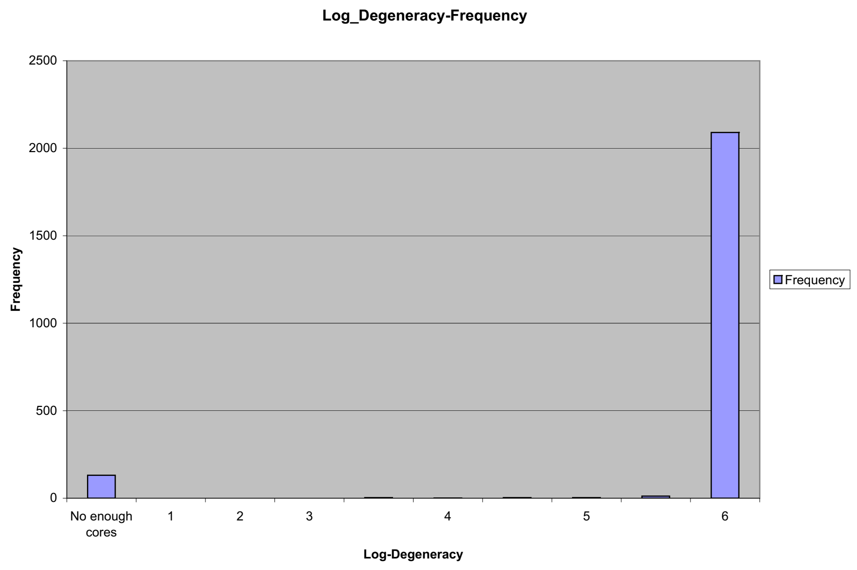


Figure 6.4: This figure shows the frequency of sequences according to the logarithmic scale of the degeneracy from the same date of the previous figure.

to outside the core (This might decrease in many cases the density and increase the possibility of increasing the solutions). From this statistics we can even consider a sequence with 10^5 as protein like sequence. However our aim and motivation is still around finding sequences with really reasonable small amount of solutions. In the next section we will talk about producing a protein like sequences, with really small amount of solutions. The degeneracy in case of FCC lattice is even much more. According to similar statistics that was made to the cubic lattice, the sequences that have solutions more than a million in our model is 100% (for more than 4000 random generated sequences). Therefore we will like to show some statistics on smaller sequences for FCC to see how the degeneracy problem appears here powerfully. in figure 6.5 we made some statistics upon short sequences (10 amino acids) to see the degeneracy of our model in FCC. Although the sequences length is less than half that in the previous experiment of the cubic lattice the degeneracy continued to be very high (if the length was equal 27, it will be much more higher in the FCC lattice than in the cubic). The figures shows that about 95% of the sequences have degeneracy of more than 10^6 while only 0.3% have degeneracy between 10^5 and 10^6, and only 0.15% have degeneracy between 10^3 and 10^4, and about 5% of the sequences do not have enough cores in the current database. To understand why the degeneracy in FCC is even much higher than in cubic lattice we have to know that the cores are too many, as well as the degree of freedom to place the backbones is too high in the side chain-backbone model. For example if we have a sequence of length 10 with H's only (the side chains should all be fit in the H-Core), we will get already 1784 solutions, and by changing one H by P to get "HHHHHHHPHH" we will get 907898 solutions. This can be an idea of the big degeneracy we get in both Cubic and FCC lattices in the new model of side chain-backbone, and we can see that the degeneracy is much more than in the case of the simple monomer model in [27] although it was already high. The figure 6.6, shows the degeneracy in the simple backbone model that was done by [27]. However the most important conclusion we can get from this statistics is to see how useless it is by using a statistical sub-optimal methodology , because we already can get huge amount of exact solutions without adding any level of sub-optimality, and so using a constraint satisfaction approach was the best to do with such kind of problems.

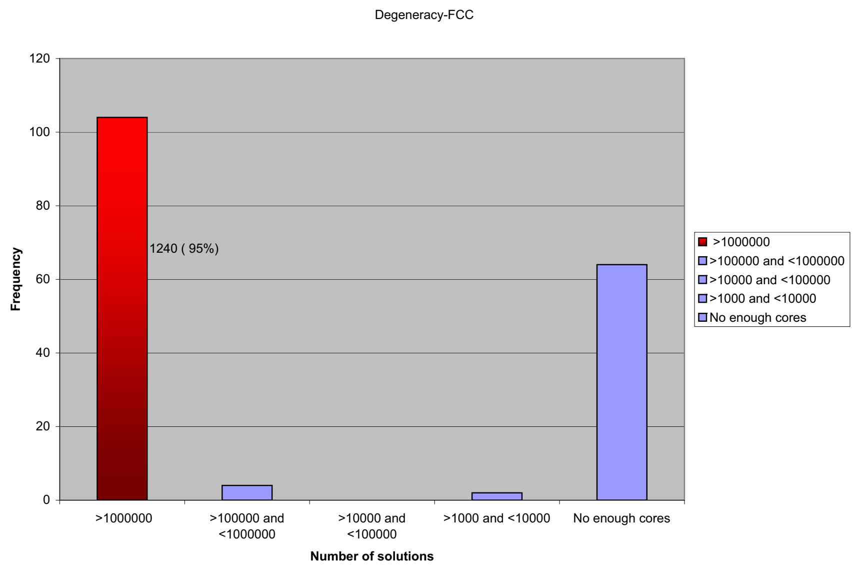


Figure 6.5: This shows the degeneracy of the sequences and their frequency. We can see the very high value of degeneracy, which is expected

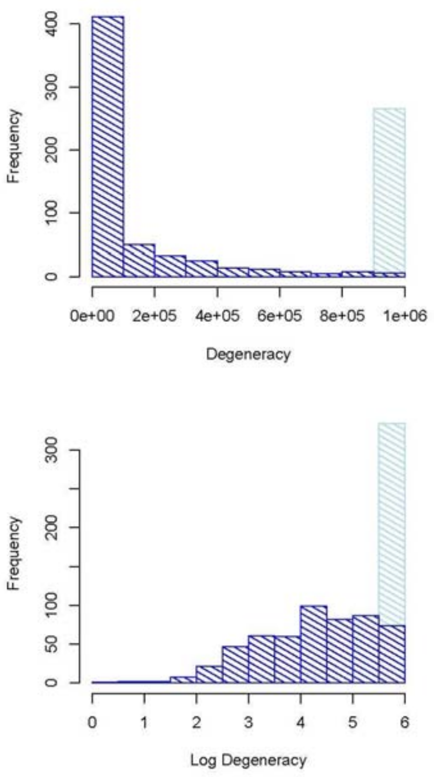


Figure 6.6: The figure shows the degeneracy in the backbone model, done by [27], also for sequences of length 27 in the FCC lattice.

6.5 Generating Protein Like Sequences

As we explained, the degeneracy is pretty high in the side chain-backbone model and even much larger than the simple monomer protein model. Thus finding sequences with low degeneracies is a hard task. We will call the sequences that have low degeneracies (lower than a specific threshold) a protein like sequences. It is important to mention that a sequence could be protein like in a simple backbone protein model but though not necessary protein like in our side chain- backbone model. Also it could be protein like if the lattice to be occupied is a Cubic lattice but though not necessarily a protein like in FCC. Trying to get sequences that have a reasonable amount of solutions among this huge degeneracy we used the Monte Carlo algorithm, just like the study in [27]. We give here a quick explanation of the Monte Carlo algorithm. We make simulation run for protein structure prediction. However we run the protein structure prediction in the silent mode, because in our case we are not interested to know the solutions but rather to know how many solutions do we get. We specify a relatively low degeneracy, and so we begin with sequences that have at least as low as this degeneracy. According to our experiments in the previous section of the delegacies statistics we already know that choosing for example a degeneracy with 10^5 is already a good start. If we already know a starting sequence with 10^4 this will be also a good start. So we can begin with any starting sequence with $deg(seq) < t$ where t=100000. We apply the Monte Carlo step on the chosen (or given) starting sequence which is:

1. **Apply random mutation to one point:** From H to P or vice versa in any letter in you sequence seq_{step}, we call the resulted sequence seq_{new}.
2. **Compute the new degeneracy:** Compute the degeneracy of your new sequence $deg(seq_{new})$.
3. **Get the new sequence:** Assign the value for the seq_{step+1} as following:
   if $deg(seq_{new}) < (deg(seq_{step})$ then $seq_{step+1} = seq_{new}$,
   otherwise $seq_{step+1} = seq_{step}$ with very high probability but we allow the assignment $seq_{step+1} = seq_{new}$ with the probability value (cut-off)
   $e^{\frac{lndeg(seq_{new} - lndeg(seq_{step}))}{-T}}$.

Sequence	Number of solutions
HPHHPPHHPHHHPHHPHHHPPHHHHPHH	7056
HPHHPPHHPHHHPHHPHHHPHHHHHPHH	1764
HPHHPHHHPHHHPHHHHHHPHHHHHPHH	1024
HHHHHHHHHHHHPHPHHHPPHHHHHPHH	792
HHHHHHHHPHHHPHHHHHPPHHHHHPHH	576
HHHHPHHHHPHHHHHHHHHHHHHHPPHH	408
HHHHPHHHHPHHHHHHHHHHPHHHHPHH	342
HHHHHHHHHHHHPHHHHHPPHHHHHPHH	288
HHHHHHHHHHHHPHHPHHPPHHHHHPHH	288
HHHHPHHHHPHHHHHHHHHHHHPHHPHH	144
HHHHPHHHHPHPHHHHHHHHPHHHHPHH	48

Table 6.4: Protein like Sequences for cubic lattice with length 28

We repeat the Monte Carlo step till we get the wanted threshold or the iterations reached the maximum limitation. After we applied the Monte Carlo algorithm to 28 length sequences we could get many protein like sequences for the cubic lattice, although as we've seen before this task seems to be pretty hard according to the statistics given in the previous section. We have here to assure that all shown numbers of solutions represent the solutions number after breaking the symmetry. The table 6.4 shows some good result, and even we reached a sequence which has only 48 solutions (that was after many iterations of the Monte Carlo step, and we chose only to see some of the protein-like sequences). It is not intuitive to get an estimation of the degeneracy of a sequence, and a good example of this is the sequence "HPHHPPHHPHHHPHHPHHHPPHHHHPHH". This sequence has only 7046 solutions however replacing only one hydrophobic amino acid into a polar will change this, so the sequence "HPHHPPHHPHHHPHHPPHHPPHHHHPHH" has degeneracy over the 10^6. This is not so intuitive, however can be explained as going one level below of optimality (The cores contacts became lower) and so the number of cores is increased , as well as the many degrees of freedom created by this polar monomer.

Chapter 7

Conclusion and Discussion

7.1 Conclusion

After looking at the obtained results by implementing the constraint satisfaction approach for side chain HP model, we can conclude that this approach allowed us to get practical exact prediction for the side chain model, which is biologically more convenience than a representation with backbones only. Also the statistics which was given about the degeneracy of the side chain model give us a very strong reason to adopt the constraint approach. The large space of exact solutions which are obtained in most cases gives no meaning when using an approach that offers approximation instead of exact optimality. The results showed also that a protein like sequence in a model without side chain is not necessarily a protein like in the side chain model. Other nonintuitive remark is that a minor change in a give sequence (may be one letter mutation) could dramatically change the size of the solutions space. Those briefly mentioned conclusions may increase our understanding to the protein prediction problem and contribute to solve the remaining open problems.

7.2 Future work

The current work could be extended to solve more open problems. On of the possible areas for such extension, is to extend this constraint approach for side chain HP model to extended alphabets like HPNX-Model (This model was introduced in [19]) but with the consideration that the HP is with side chains. Also one of the future possible work is to compare the results obtained by a model that simulates the protein folding in side chain HP model , with our model that predicts the exact solutions that match the stable thermodynamics state of minimal energy. Also more efficiency is always in need, and coming up for example with a parallel model of constructing the HCore will be a good idea, especially that a simple implementation is already done to be able to distribute the calculation on several clusters.

Bibliography

[1] Modeling evolutionary landscapes: mutational stability, topology, and superfunnels in sequence space. *Proc. Natl. Acad. Sci. USA*, (96(19)):10689–94, 1999.

[2] A Sali A. R Dinner and M Karplus. The folding mechanism of larger model proteins: role of native structure. *Proc. Natl. Acad. Sci.USA*, (93(16)):8356–61, 1996.

[3] Rolf Backofen and Sebastian Will. Fast, constraint-based threading of hp-sequences to hydrophobic cores. In *Lecture Notes in Computer Science*, pages 494–508. 7th International Conference on Principle and Practice of Constraint Programming (CP'2001), 2001.

[4] Rolf Backofen and Sebastian Will. Approximation algorithms for protein folding prediction. In *Constraints*, 7(3), pages 333–349, 2002.

[5] B. Berger and T. Leighton. Protein folding in the hydrophobichydrophilic (hp) model is np-complete. *Journal of Computational Biology*, pages 27–40, 1998.

[6] F. E. Cohen and S. R. Presnell. The combinatorial approach, in protein structure prediction. *M. J. E. Sternberg, Oxford University Press,Oxford*, pages 207–228, 1996.

[7] K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, (24):1501 1509, 1995.

[8] R. F. Doolittle. *Computer Methods for Macromolecular Sequence Analysis.* Academic Press, San Diego, CA, 1996.

[9] G. J. B. Williams E. F. Meyer M. D. Brice J. R. Rodgers O. Kennard T. Shimanouchi F. C. Bernstein, T. F. Koetzle and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures.

[10] Volker Heun. Approximate protein folding in the hp side chain model on extended cubic lattices. In *Computational molecular biology series issue IV*, volume 127, pages 163 – 177. Elsevier Science Publishers B. V., 2003.

[11] P. D. Thomas H. S. Chan E. I. Shakhnovich K. Yue, K. M. Fiebig and K. A. Dill. A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci*, 92(325-329), 1995.

[12] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica, Vol.28*, pages 497–520, 1960.

[13] Martin Mann, Sebastian Will, and Rolf Backofen. CPSP-tools - exact and complete algorithms for high-throughput 3D lattice protein studies. *BMC Bioinformatics*, 9:230, 2008.

[14] Mauri, Pavesi, and Piccolboni. Approximation algorithms for protein folding prediction. In *Symposium on Discrete Algorithms*, pages 945–946. A Conference on Theoretical and Experimental Analysis of Discrete Algorithms, 1999.

[15] C. Papadimitriou A. Piccolboni and M. Yannakakis P. Crescenzi, D. Goldman. On the complexity of protein folding. *Journal of Computational Biology*, pages 61–62, 1998.

[16] Arnold L. Patton, W. F. Punch III, and E. D. Goodman. A standard ga approach to native protein conformation prediction. In Larry Eshelman, editor, *Proceedings*

of the Sixth International Conference on Genetic Algorithms, 7(3), pages 574–581, San Francisco, CA, 1995. Morgan Kaufmann.

[17] E. Bornberg-Bauer R. Wroe and H. S. Chan. Comparing folding codes in simple heteropolymer models of protein evolutionary landscape: Robustness of the superfunnel paradigm. *Biophysical*, (88):118–131.

[18] Régin. A filtering algorithm for constraints of difference in csps. AAAI, 1994.

[19] Sebastian Will Rolf Backofen and Erich Bornberg-Bauer. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, (15(3)):234 – 241, 1999.

[20] B. Rost and C. Sander. Structure prediction of proteins–where are we now? *Opin.Biotechnol.*, (5):372–380, 1994.

[21] B. Rost and C. Sander. Bridging the protein sequence-structure gap by structure predictions. *Annu. Rev. Biophys.Biomol. Struct.*, (25):113–136, 1996.

[22] Jaramillo A. Wernisch L. Hery S. and Wodak S.J. Native protein sequences are near optimal for their structures in the protein core but not on the surface. *Proc. Natl. Acad. Sci. USA 99(21)*, pages 13554–13559, 2002.

[23] E. I. Shakhnovich and A. M. Gutin. Enumeration of all compact conformations of copolymers with random sequence of links. *Journal Chemical Physics*, (8):5967–5971, 1990.

[24] M.J. E. Sternberg. *Protein Structure Prediction.* Oxford University Press, Oxford, 1996.

[25] Ron Unger. The genetic algorithm approach to protein structure prediction. In *Structure and Bonding*, volume 110, pages 153 – 175, 2004.

[26] Sebastian Will. Constraint-based hydrophobic core construction for protein structure prediction in the face-centered-cubic lattice. *Bioinformatics Journals Computational Biology*, (April 03):661 – 672, 2002.

[27] Sebastian Will. *Exact, Constraint-Based Structure Prediction in Simple Protein Models.* PhD thesis, University of Freiburg, 2005.

[28] Kaizhi Yue and Ken A. Dill. Sequence-structure relationships in proteins and copolymers. *Phys.Rev. E*, pages 2267–2278, 1993.

Selbständigkeitserklärung

Hiermit erkläre ich, dass die hier vorliegende Masterarbeit von mir selbständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen erstellt wurde.

Freiburg, den

Unterschrift