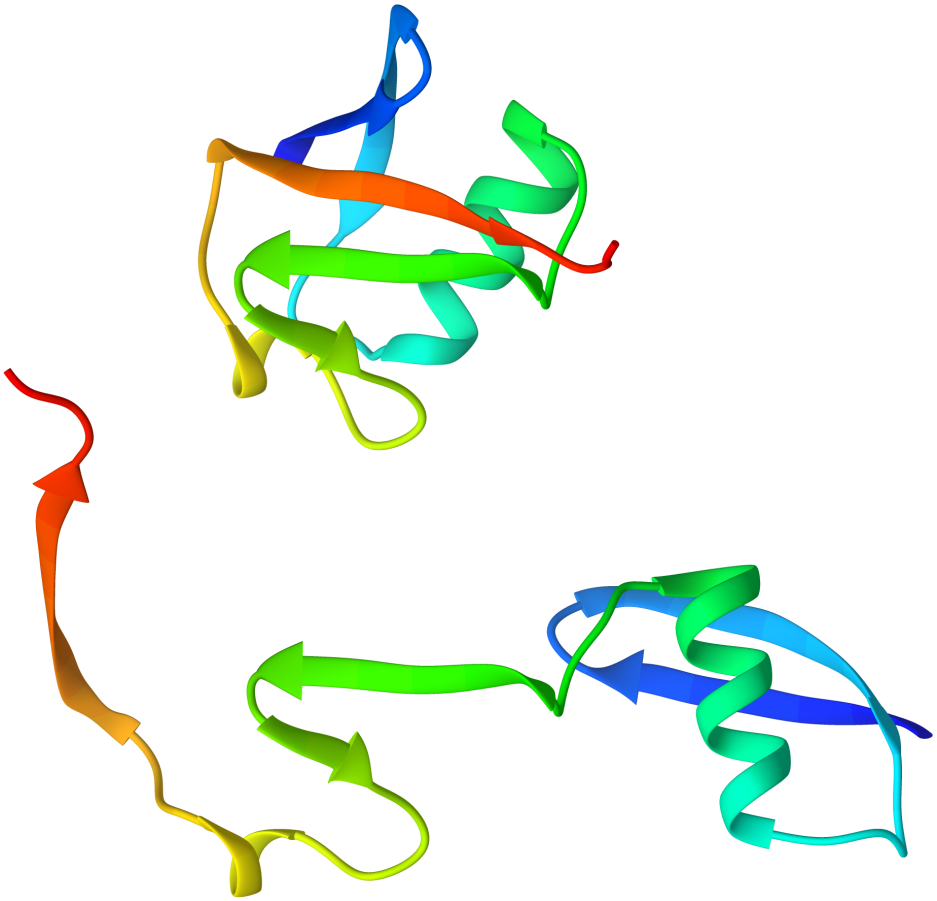

Visualising protein unfolding pathways

Master's thesis

MIHAIL ANTON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Visualising protein unfolding pathways

MIHAIL ANTON

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Visualising protein unfolding pathways
MIHAIL ANTON

Supervisor: Graham Kemp, Department of Computer Science and Engineering
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Ubiquitin in native state next to an intermediate state of the computed unfolding pathway.

Typeset in LaTeX
Gothenburg, Sweden 2017

Visualising protein unfolding pathways
Mihail Anton
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Observing interactive unfolding animations of proteins improves the understanding of protein chain movements. This work focuses on how the movements can be computed using a reduced conformational space, made of mesostates. To this end, the present work has implemented through constructive research an extension of a popular molecular visualisation software. The extension uses beam search to generate a motion graph from which an unfolding pathway is extracted with a shortest-path algorithm. The unfolding animation is obtained by morphing the protein chain through the pathway's conformations. The results for unfolding ubiquitin resemble, to some extent, the reverse of the folding through molecular dynamics. Results also show that different parameters are needed for unfolding a knotted protein and β-sheet rich proteins. The extension shows that the protein main chain movements can be restricted to mesostates. The extension also proved to be helpful in discussing unfolding functions and provides a platform for verifying such functions.

Keywords: protein, unfolding, pathway, visualisation, animation, motion planning, beam search, mesostates.

Acknowledgements

"Live as if you were to die tomorrow. Learn as if you were to live forever."
— Mahatma Gandhi

Here I am, at the end of this project, through the support of many. I deeply appreciate the patience, encouragement and guidance I have received. I will remain forever thankful to my fountain of joy Marina, my amazingly strong sister Alexandra and my loving parents Ana and Felix for being there for me. I am also thankful to my friends Silvia and Cristina for showing me what confidence is; to My for the long walks; and to Khaled for teaching me to cherish life, as imperfect as it is. This work has come to fruition thanks to Graham, who was always available for a chat.

Mihail Anton, Gothenburg, July 2017

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Aim . . . 2
1.3 Overview . . . 2

2 Background on proteins **3**
2.1 Amino acids . . . 3
2.2 Protein structure . . . 5
2.3 Protein chain dynamics . . . 7
2.4 Protein folding and unfolding . . . 7

3 Methods **9**
3.1 Motion planning . . . 9
3.2 The protein model . . . 10
3.2.1 Mesostate alphabets . . . 11
3.2.2 Mesostate alphabet expansion . . . 12
3.2.3 Approximation to mesostates . . . 13
3.2.4 Mesostate hierarchy . . . 13
3.2.5 Rigid and flexible amino acids . . . 14
3.3 Scoring function . . . 15
3.4 Collision detection . . . 15
3.5 Graph exploration . . . 16

4 Implementation **19**
4.1 Visualisation – Chimera . . . 19
4.2 Motion planning – `graph-tool` . . . 19
4.3 Development setup . . . 20

5 The `Unfolding` extension **21**

6 Results **23**
6.1 Studied proteins . . . 23
6.2 Unfolding 1UBQ . . . 24

6.3 Unfolding 2EFV 26
6.4 Further evaluation 27

7 Discussion **31**
7.1 Mesostates 31
7.2 Scoring function 32
7.3 Collision detection 33
7.4 Graph exploration 34
7.4.1 Improvements 34

8 Conclusions **37**

Bibliography **39**

A Appendix 1 **I**

List of Figures

2.1 Three consecutive amino acids, with main chain atoms coloured in shades of green, side chains coloured in shades of blue and bonds defining the phi and psi angles. 3

2.2 The Ramachandran plot, obtained with UCSF Chimera, for a protein called ubiquitin. The dots correspond to the phi and psi values of all amino acids. 4

2.3 Ubiquitin in multiple visualisation formats: atom centres, atom spheres, protein surface. The dominant representation is a rainbow coloured ribbon. All colouring is based on conventions. 5

2.4 Protein main chain in green, with transparent side chains, forming an α-helix, seen from both ends. Edited from PDB entry 1CRN. 6

2.5 Protein main chain in green, with transparent side chains, forming a β-sheet. Edited from PDB entry 5HBS, by removing the loops that connect the β-sheets. 6

3.1 The one-bead model, where an amino acid has only two variables: phi and psi. 11

3.2 The Ramachandran plot of the 11 mesostate alphabet by Chellapa and Rose (2012). 12

3.3 The expansion of the 11 mesostate alphabet, shown as red plus signs. 13

3.4 The undirected mesostate transition map, with a maximum node degree of 5. 14

3.5 Example of beam search with size 3. On each layer, only 3 nodes in purple are explored further. 17

5.1 The `Unfolding` extension implemented in Chimera. 21

6.1 The flexible amino acids for 1UBQ highlighed in red, indexing from position 0: 6-9, 17-21, 34-38, 44-47, 50-54, 59-63. Only about 37% of the protein's chain is flexible. 24

6.2 The expansion of the unfolding graph for ubiquitin, in a radial tree layout. The explored nodes are highlighted in cyan. The expanding radii correspond to the tree levels, as created by the beam search exploration. In step 1, only five mesostate changes can be made from the starting native conformation: mesostate G for amino acid 60, B for 61, P for 62, U for 9 and P for 36. The small number of possible mesostates is because the phi and psi angles first need to be approximated to a mesostate before transitions to other mesostates can be made. The small letters on the labels indicate that the phi and psi angles at those positions have the original values, and to which mesostate they are closest to. In step 2, all 5 mesostates are accepted by the beam and are consequently explored. The top scoring 6 mesostates are further explored in step 3 and so on. 25
6.3 The final graph for 1UBQ. The circles highlight the tree exploration iterations, with the start node in the centre. All nodes are clash-free mesostate transitions. The nodes that are discarded at each iteration are in yellow. The nodes in cyan give the unfolding motion; they are retrieved by Dijkstra's algorithm. 26
6.4 The unfolding animation of 1UBQ obtained with the `Unfolding` extension implemented in Chimera. 27

A.1 Start and end conformations of the unfolding animation of 1HRC. . . I
A.2 Start and end conformations of the unfolding animation of 1LMB. . . I
A.3 Start and end conformations of the unfolding animation of 1SHF. . . II
A.4 Start and end conformations of the unfolding animation of 2ABD. . . II
A.5 Start and end conformations of the unfolding animation of 2PTL. . . III

List of Tables

6.1 Evaluation of the `Unfolding` extension of 12 topologically diverse proteins. The success rate for this set is approximately 50%, which is not surprising given the very reduced conformational space. 28

1

Introduction

Proteins are important biological macro-molecules – they are an essential part of a living organism's cells and are required for the function and regulation of the body's tissues and organs. Many diseases are known to be caused by malfunctioning proteins (Dobson 2003). Therefore, understanding all aspects related to proteins is significant for improving health care.

The protein lifetime can be loosely divided in several stages. Through synthesis, a protein is created. Then, it undergoes assembly. Having reached the desired shape of operation, it performs its function. Finally, through a disassembly process, the protein's components end up being reused. It is well established that the protein motion is reliable and mechanistic, both during assembly and disassembly. Therefore, it is important to understand the mechanics of proteins in a concrete way – through visualisations.

1.1 Motivation

The computed motion of proteins has already been turned into videos. These are usually obtained through computer molecular dynamics simulations that take significant computation time on dedicated machines, even for small proteins. Since the simulation is based on calculating what happens with each atom in a volume that contains water and a protein, there is significant motion at the atomic level. This makes the large-scale motions of the protein hard to follow. Therefore, the usefulness of videos is limited by one's capacity of abstracting the atomic vibration. The videos could be improved by further refining the protein's motion. Obtaining an easy to follow video requires a fair amount of computer time, human time and specialised knowledge. Such efforts limit the number of proteins for which videos are made.

There are better ways of visualising protein motions than videos. Because of the nature of videos, the viewing angle is fixed. In contrast to videos, a molecular visualisation software offers full control over the viewing, including viewing angles and molecular representation.

For protein mechanics, prototyping is useful. Over the decades, researchers have proposed many functions that guide the protein motion. However, these have not been visually compared. Moreover, many functions operate on a higher level of abstraction than that of the atoms, which means they can not be implemented in molecular dynamics simulations.

Therefore, it would be useful to have a way to compare existing functions that guide the protein motion with the help of molecular visualisation software.

1.2 Aim

In this work, the aim is to create a solution to visualise protein motion on the computer, specifically the disassembly of proteins, which is simpler than the assembly, but follows the same basic rules. The implementation will model large scale movements of a protein's motion, starting from the native conformation. Besides facilitating the creation of animations of proteins unfolding, it should enable the testing of different functions that guide the protein's motion. This work is aimed at improving the understanding of the protein mechanics, by providing a way to see the effects of various unfolding functions. Moreover, the work is set to use and implement algorithms to answer the following research question:

How can a protein chain be animated using constrained motion?

1.3 Overview

This thesis starts by introducing the basic concepts of proteins in chapter 2: how amino acids make up the protein chain and how they determine the protein structure and chain dynamics. Then, the folding and unfolding areas are introduced.

In chapter 3, Methods, the thesis continues by describing the protein model and the algorithms relevant for this work and why they were chosen for the implementation. Changes to the algorithms are also presented and the new contributions of this work are detailed.

Next, chapter 4 describes the essential programming dependencies of this project, their advantages and disadvantages.

The outcome of the implementation, the Chimera extension, is presented in chapter 5. The thesis continues by showing the results of unfolding several proteins with the extension in chapter 6.

Different challenges and improvements to the methods are discussed in chapter 7. Lastly, the work is summarised in a concluding chapter.

2

Background on proteins

Proteins are highly complex molecular structures. This chapter introduces the area of structural bioinformatics concerned with protein folding, explaining the fundamental concepts of protein structure and motion.

2.1 Amino acids

Proteins are made of protein chains – biological polymers that can be constructed from about 20 different amino acids. The length of the chain spans from the lower tens to tens of thousands of amino acids.

The protein chain is encoded as a string of capital letters, where each letter uniquely identifies an amino acid. This string has an associated order; a reverse sequence identifies a distinct protein. This order is defined by the type of atoms that are first and last in the protein chain: the nitrogen atom at the beginning and the carbon atom at the end. A protein's sequence encodes only its amino acid composition. A protein's unique sequence of amino acids is always associated with the same structure. However, the same overall structure can be obtained from different protein sequences. Therefore, when looking for a specific protein structure, a structure encoding sequence is more reliable than an amino acid encoding. The Methods section describes how this work uses a method of encoding the structure of a protein.

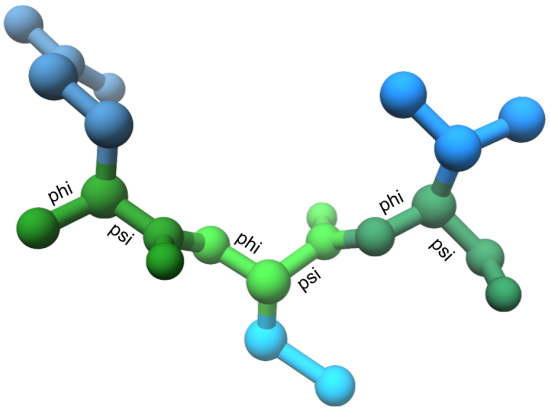

Figure 2.1: Three consecutive amino acids, with main chain atoms coloured in shades of green, side chains coloured in shades of blue and bonds defining the phi and psi angles.

An amino acid is composed of a number of atoms. All atoms are connected to other atoms through chemical bonds. Some of the bonds are internal – between atoms of the same amino acid; other bonds are created by an atom from one amino acid with an atom from a consecutive neighbouring amino acid, thus composing the main chain. Figure 2.1 illustrates the main chain and the side chain atoms.

In each of the amino acids there can be rotation about two main chain bonds, designated phi and psi. The values of the phi and psi angles measure the rotation of the bonds. The pairs of phi and psi of all amino acids are sufficient to describe the structure of the main chain. Figure 2.2 shows an orthogonal distribution of the phi and phi values in a representation called the Ramachandran plot.

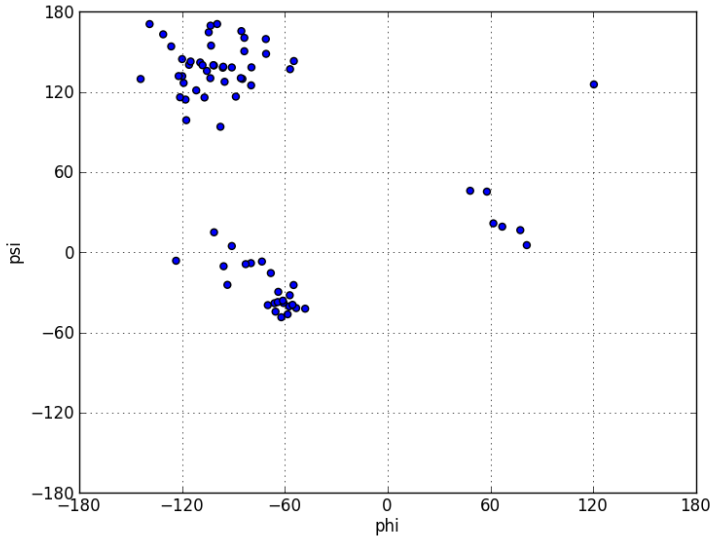

Figure 2.2: The Ramachandran plot, obtained with UCSF Chimera, for a protein called ubiquitin. The dots correspond to the phi and psi values of all amino acids.

Some atoms of an amino acid are not involved in keeping the protein chain connected; these atoms make up the side chain. One common side chain property is the degree of repelling or attracting water molecules that usually surround proteins. Some side chains are strongly hydrophobic - they prefer to sit together, isolated from the water molecules by other atoms of the protein. Other side chains are hydrophilic - they prefer to be in contact with water molecules. There is a number of side chains for which hydrophobic/hydrophilic preference is weak. The preference can also depend on the neighbouring atoms.

An amino acid can have more side chain atoms than main chain atoms. When manipulating the visualised protein chain by changing the phi and psi angles, if side chains are not flexible atoms often end up overlapping in 3D space. The superposition or collision of atoms is called steric clash. Avoiding steric clashes is one of the fundamental rules in protein folding.

2.2 Protein structure

The structure of a protein has 4 levels of abstraction. These can be seen as equivalent to a zoom level – their granularity makes each level adequate for certain uses. In this work, only the primary and secondary structures are relevant. Figure 2.3 shows multiple protein visualisation formats, highlighting different structural levels.

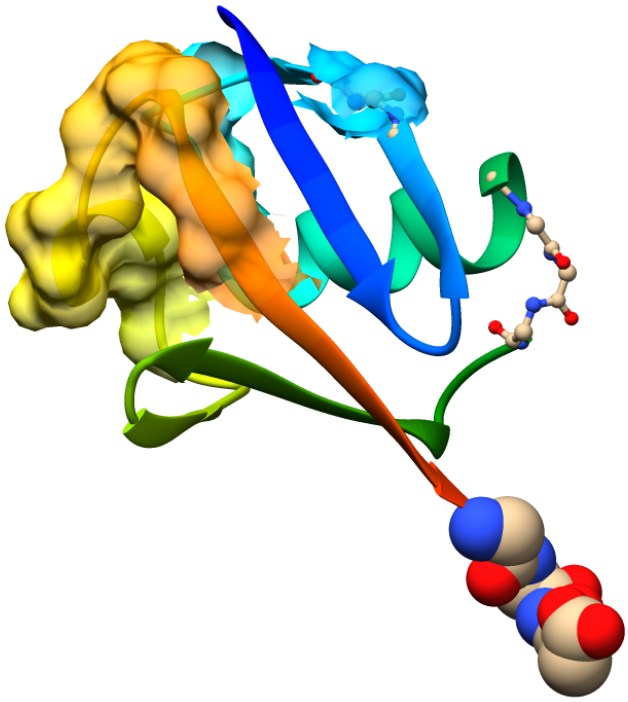

Figure 2.3: Ubiquitin in multiple visualisation formats: atom centres, atom spheres, protein surface. The dominant representation is a rainbow coloured ribbon. All colouring is based on conventions.

The primary structure is the most detailed level. Each atom is individually positioned and represented, with the exception of hydrogen atoms, because of their high degree of positional flexibility. In order to simulate the unfolding, an algorithm operates on the primary structure, because of how the phi and psi angles are defined by the chemical bonds between atoms.

The secondary structure is composed of small sets of consecutive amino acids forming a particular shape. The length of sets belonging to a secondary structure is typically between 8 and 12, and rarely more than 20 amino acids. In an α-helix, the main chain is wrapped around an imaginary cylinder, as shown in Figure 2.4. In a β-sheet, the atoms lie flat, as shown in Figure 2.5. Both structures are stabilised by regular, repetitive patterns of hydrogen bonds (Baker and Hubbard 1984); thus, they are more rigid than random structures. One study showed that the formation of helices generally precedes sheets in the folding process (Kubelka et al. 2004). Moreover, almost identical sequences of amino acids can be part of both helix and sheet structures (Kabsch and Sander 1984; Porter et al. 2015).

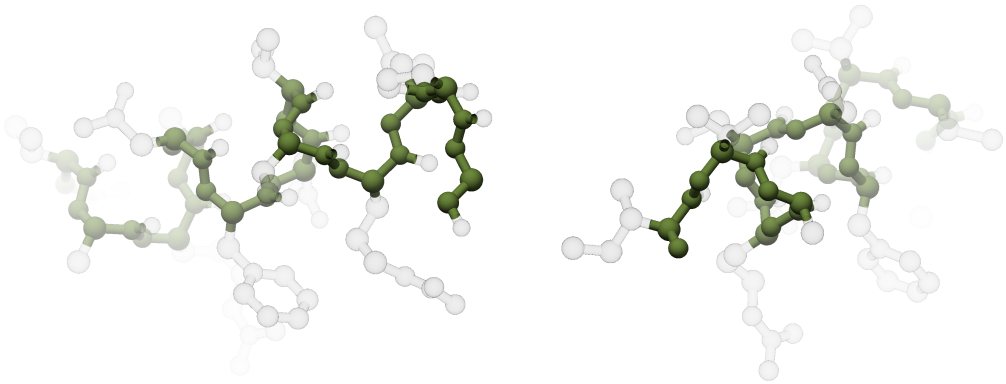

Figure 2.4: Protein main chain in green, with transparent side chains, forming an α-helix, seen from both ends. Edited from PDB entry 1CRN.

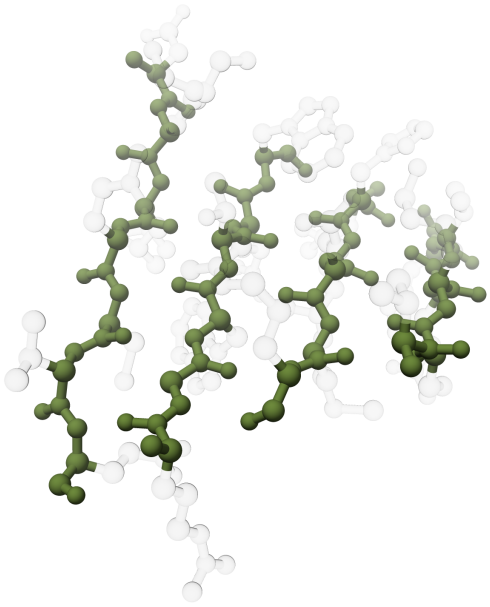

Figure 2.5: Protein main chain in green, with transparent side chains, forming a β-sheet. Edited from PDB entry 5HBS, by removing the loops that connect the β-sheets.

Secondary structures can be classified with the help of algorithms. One such established algorithm is DSSP (Kabsch and Sander 1983), but there are others. In molecular visualisation software, there exist special depictions that make the secondary structures stand out from other parts of the protein.

The secondary structure level also has other patterns: hairpin, loop, turn and coil. They are the more flexible parts of the protein and make a good target for the amino acids from which the protein should unfold.

2.3 Protein chain dynamics

A fundamental concept in bioinformatics is that the amino acids of a protein chain are always bonded to each other. This means that the chemical bonds in the main chain are not spontaneously broken and reformed – the main chain cannot pass through itself. This is particularly relevant for the class of knotted proteins, where by imaginary pulling the two ends of the protein chain would create a knot. Even in the formation of the knot, the chain cannot pass through itself (Taylor 2000). Knotted proteins are important to be studied, because even though their dynamics must obey the same rules as typical proteins, the motion is unlike a typical protein; the folding order is more strict. Many knot topologies exist; in some cases the knot is particularly deep (Taylor 2007), which challenges the understanding of protein chain dynamics.

Proteins have folded and unfolded states. Although atoms always vibrate, a protein's folded state is singular. The unfolded state is, on the contrary, not singular. Rather, as defined by Karplus et al. (1995), it is the set of states where the protein chain is lacking a stable 3D structure and many stabilising contacts. The unfolded state is often called random, although this is a misnomer (Plaxco and Gross 2001).

The protein unfolding animation is obtained by the rotation of the chemical bonds in the main chain. Unfolding has to start from the folded conformation by rotating the bonds until it reaches an unfolded conformation. Molecular visualisation software can perform the animated rotation of chemical bonds in real time.

2.4 Protein folding and unfolding

Proteins naturally adopt a defined shape to perform their function. Under physiological conditions, a protein chain spontaneously adopts a native conformation, or simply, a native 3D shape. This discovery was awarded a Nobel Prize in 1972 (Nobel Media AB n.d.). One exception is the class of intrinsically disordered proteins that rely on a particular environment to adopt the conformation that fulfils the desired function (Wright and Dyson 1999). Only about one third of eukaryotic proteins are intrinsically disordered (Ward et al. 2004). Regardless of their class, all proteins have evolved to consistently adopt a conformation related to their function.

The transition process from a random structure to a native conformation is called folding and is the result of rotating the chemical bonds in a protein's chain.

While research in the field of protein folding has been ongoing for over 50 years (Dill and MacCallum 2012), predicting the conformation of a protein chain given the sequence of its building blocks remains one of the most important and difficult problems in bioinformatics. This problem does not become easy to solve even when the native conformation is experimentally determined.

While folding is conceived as a funnel (Onuchic et al. 1995), unfolding is the opposite. When folding, given any starting 3D shape and an adequate environment, the protein will achieve its native conformation. When unfolding, the motion starts

from the native conformation and ends when the protein chain is relatively elongated – visually similar to a loose chain. While the native conformation is generally fixed and can often be determined experimentally or computationally, less is known about the unfolded conformation. In addition, the unfolding motion "is not always a direct reversal of the folding process" (Chan and Dill 1998).

There has been work on the computational simulation of folding for about fourty years (Levitt and Warshel 1975). The intrinsic computational difficulty limits the simulation of all but the fastest folding proteins (Freddolino et al. 2010; Mayor, Guydosh, et al. 2003). The unfolding process is, on the other hand, easier to tackle. Unlike the folding process, where the target is described by a single state, the unfolding process can terminate in many different valid conformations. In molecular dynamics simulations, unfolding can be computationally fast if the temperature factor is set very high, for example above 200°C (Fersht and Daggett 2002), without affecting the unfolding pathway (Mayor, Johnson, et al. 2000). In practice, this would be equivalent to heating up the protein solution. Animations obtained from molecular dynamics simulations, however, require further processing to filter out the inherent atom vibrations, while keeping the ample movements.

Motion planning theory, commonly used in robotics, has been used in identifying protein folding pathways, due to the similarity between the joints of robotic arms and the “joints” in the protein chain. The algorithmic part of this thesis is inspired by the use of motion planning on folding pathways.

3

Methods

Algorithms have been studied for their applicability to the protein folding problem for more than four decades. The problem was tackled from two sides: reducing the search space and using an efficient algorithm.

The vast search space justifies the selectivity of the folding process. If 100 amino acid proteins were to sample all possible 3D shapes defined by the phi and psi angles of each amino acid, "the universe would end before chains could encounter the native conformation via an unguided search" (Rose et al. 2006). Studies show that proteins fold at a microsecond to millisecond timescale (Myers and Oas 2002 as cited by Rose et al. 2006). The argument for selectivity remains valid even when the aim is to unfold a protein – the unfolding process is selective, too.

Many researchers have worked with simplified protein representations. For example, a simplified model of protein structure and a reduced degree of motion freedom were used in a computer simulation over four decades ago (e.g. Levitt and Warshel 1975). Other common simplifications include creating a conformational search space by restricting the amino acids in a protein chain to lie on nodes in a regular lattice (Lau and Dill 1989).

In this work several established algorithms will be implemented and applied to obtain a protein unfolding pathway. The implementation will also use a set of simplifications that have not been previously used in conjunction with the chosen algorithms.

This chapter will present the work, starting with an overview of motion planning algorithms and the protein model used in this work. Following is an overview of the applied algorithms, where the scientific context will be contrasted against the current work. The chapter ends with comments on the implementation.

3.1 Motion planning

The current work focuses on the unfolding of proteins, inspired by the application of motion planning algorithms to protein folding. It is probably the similarity of the protein chain to a multi-joint robot arm that has sparked the use of this algorithm. The main chain motion taking place at a single amino acid can be abstracted to the rotation of a joint.

A recent review (Al-Bluwi et al. 2012) compares how two classes of motion planning algorithms, Probabilistic RoadMap (PRM) and Rapidly-exploring Random Tree (RRT), are used to obtain the folding motion given by the shortest path in a motion

graph. A* or Dijkstra's algorithm can be used to retrieve the shortest path which describes the step-by-step conformations that make the motion.

Both motion planning algorithms rely on a step that involves randomness to expand the underlying graph data structure. In PRM, a random sample is chosen from the configuration space; if valid, it is added to the graph. In RRT, the tree is expanded from the start node towards a valid random sample.

Both algorithms expect start and end configurations. While this is an obvious element of planning the motion of a robotic arm, the same cannot be said about protein motion. The native conformation of a protein, when the protein chain is folded, can be used as a stable configuration either at the start or the end of a motion planning pathway. However, the unfolded conformation is not unique. Ideally, the chosen unfolded conformation would have to be part of a known unfolding pathway. Otherwise, the generated unfolding pathway might deviate from what is conceived as the realistic motion. Also, if the unfolded state is not fixed, the motion might differ between different runs of the program.

Both algorithms use an energy function. This is an essential difference between the molecular and robot motions. The likelihood of a molecule to adopt a conformation is related to the potential energy. For a robot, probably all conformations have equal probabilities. In protein motion planning, the energy function determines the edge weight in both PRM and RRT graphs.

In this work, the aim was to obtain a realistic unfolding motion. In the context of motion planning algorithms, an arbitrary unfolded conformation can be selected. The unfolding of a protein can be stopped at the state where the protein conformation is first considered unfolded. Such an approach is a direct application of the two algorithms. However, the resulting pathway would still be based on a random step, regardless of the chosen algorithm. Since the protein motion is not random, the implementation uses a scoring function to favour conformations.

This work has implemented a motion planning inspired solution to visualise the protein unfolding, which covers a previously unexplored research topic. The implemented search of the motion planning tree avoids random or brute-force searches by guiding the graph exploration with a search algorithm and a scoring function.

3.2 The protein model

The way the protein chain is modelled affects the magnitude of the search space in any algorithm. A review of coarse-grained models for proteins (Tozzini 2005) illustrates the relationship between the complexity of representation and the complexity of parametrisation for computer simulations, discussing the advantages and disadvantages of each. The relationship between protein model detail and protein model accuracy has also been studied, which concludes that the law of diminishing returns also applies to this relationship (Park and Levitt 1995).

In this work, a theoretical one-bead model was used, where an entire amino acid is reduced to a single bead having two parameters: phi and psi. The one-bead model is represented in Figure 3.1. By removing the side chains, the risk of steric

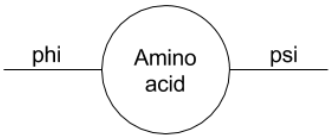

Figure 3.1: The one-bead model, where an amino acid has only two variables: phi and psi.

clashes is reduced; since no parameters were used in the motion of side chains, their presence would have led to steric overlap. From the one-bead model, the protein's entire structure can be reconstructed. The one-bead model defines the structure of the main chain and the sequence of amino acids (McCammon et al. 1980). After searching the conformational space with the main chain, side chains can be added back to the structure as a following step, as suggested by Samudrala et al. (2000).

Similar to motion planning algorithms, the one-bead model implies the protein has a rigid geometry (Al-Bluwi et al. 2012). In molecular dynamics simulations, the opposite is predicated – the vibration of the atoms drives the motion, implicitly affecting the lengths of bonds between atoms. The one-bead model is detailed enough for the computation of chain motion. However, the visual representation of the model must use all main chain atoms, in order to be able to rotate the bonds between the atoms from within the molecular visualisation software.

3.2.1 Mesostate alphabets

The one-bead model previously described can be further simplified. Existing research has found multiple ways to reduce the phi and psi search space.

To provide a different way of encoding proteins than through their amino acid sequence, Chellapa and Rose (2012) have created a solution that reduces the possible combinations of phi and psi values. An alphabet of 11 pairs of values, or mesostates, places 11 squares of 20° on the Ramachandran plot. The distribution of the centre coordinates is shown in Figure 3.2. A sequence of these mesostates, one per amino acid, describes the approximate structure of the protein, separately from the amino acid composition of the protein. The authors aimed to enable quick searches of proteins through their structure rather than amino acid sequence. In this work, mesostates are used to reduce the phi and psi search space.

A previous solution was to divide the Ramachandran plot into squares of 60° (Gong et al. 2005), obtaining 36 mesostates. In the empirical tests performed during the present work, the set of 11 mesostates performed better. Approximating a protein's structure with the 36 mesostates was less accurate than with the 11 mesostates when compared to the original structure. Unlike the arbitrary division of the 36 mesostates, the 11 mesostate alphabet is statistically selected to reflect the most common phi and psi values of a large number of proteins.

In a more recent approach, a 27 letter alphabet was generated based on the correlation with surrounding amino acids (DasGupta et al. 2015). For the proteins chosen in this work, empirical results were not superior to the 11 mesostates. Additionally,

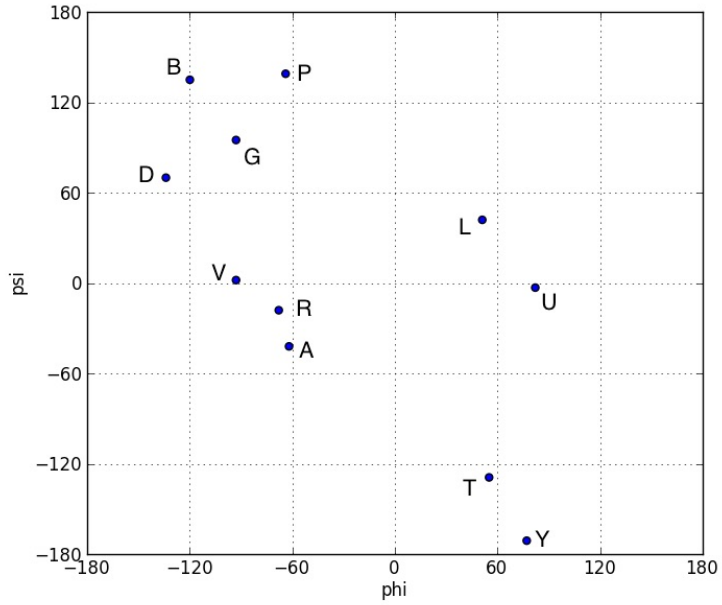

Figure 3.2: The Ramachandran plot of the 11 mesostate alphabet by Chellapa and Rose (2012).

a reduced alphabet improves the speed of calculations and does not require the use of an external solution for the approximation, in this case the authors' web service.

Some combinations of phi and psi angles are known to be impossible. This is because they would cause a steric clash either with the immediate neighbouring residues, or with the second level neighbours. Even by removing the areas of steric clashes from the Ramachandran plot, the number of possible combinations is too large to be considered efficiently; mesostates constrict the search space much more. Fitzkee and Rose (2005) show that using a 5 mesostate model eliminates 50% of the conformational space by steric clashes, when considering segments of 6 amino acids in length. However, the 11 mesostate alphabet reduces the search space to 3.4% of the conformational space. Given that this work uses only the centres of the mesostates, the conformational space used is arguably much less than even 1%.

3.2.2 Mesostate alphabet expansion

The chosen 11 mesostate alphabet was expanded in this work with intermediate mesostates (Figure 3.3). In reality, the protein folds by simultaneously altering many phi and psi pairs. Since this phenomenon cannot be represented in a motion planning approach, intermediate mesostates have been added to the Ramachandran plot. The 13 intermediate mesostates were added in order to be able to begin changing several mesostates, before either of them reaches the desired mesostate. The coordinates were chosen so as to evenly reduce the distance between pairs of mesostates, especially where the distance on the Ramachandran plot was higher than 80° .

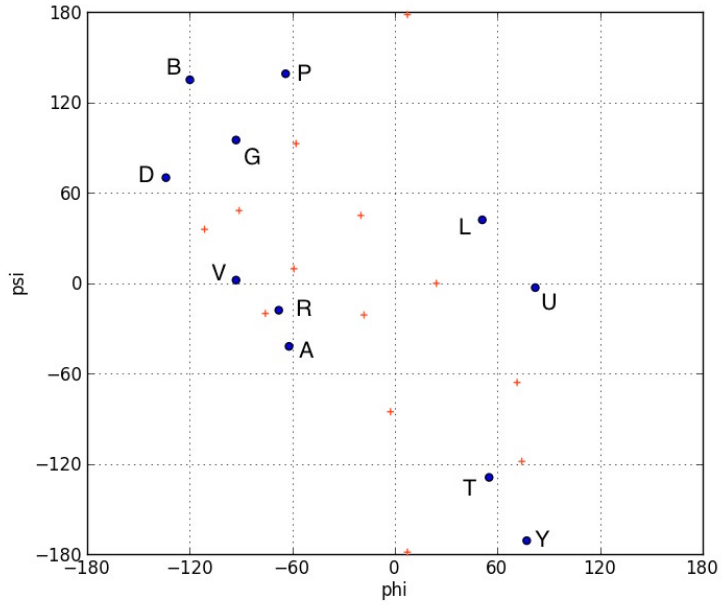

Figure 3.3: The expansion of the 11 mesostate alphabet, shown as red plus signs.

3.2.3 Approximation to mesostates

In order to make use of mesostates in the algorithm, the protein's structure needs to use the mesostate alphabet. As per the work of Chellapa and Rose (2012), the approximation was done based on each amino acid's original phi and psi values. The respective angles are approximated to the closest mesostate.

In this work, two different approaches have been used in calculating the distance between two pairs of phi and psi. In the first, the distance was calculated from the representation of two points on the orthogonal Ramachandran plot. The second works by converting the Ramachandran plot to a spherical surface. In this case, the shortest distance between two points on a sphere was calculated. It was observed that the spherical representation skewed the distance. A torus representation would also skew the distance because of how the increasing major radius stretches the orthogonal plot. Therefore, the final implementation used the orthogonal distance.

In practice, approximating an entire protein's native conformation to mesostates can generate steric clashes, even when only the main chain atoms are used in clash detection. Therefore, the start conformation of the algorithm is the native conformation, which is gradually approximated to mesostates one amino acid at a time.

3.2.4 Mesostate hierarchy

In this work, the possibility of further reducing the search space was explored. The potential hierarchy of mesostates was investigated, similar to the idea of hierarchy presented by Srinivasan and Rose (Srinivasan and Rose 1995). Assuming that a protein's chain would move only towards the desired conformation – a one-way

motion –, a hierarchy of mesostates can be imagined. For any given mesostate, any transition to a new mesostate would result in a more unfolded conformation. Results of ubiquitin unfolding gave positive results. However, this small protein cannot be considered representative of all proteins. Using a hierarchy reduces the search space, but the improvement is significant only for larger proteins, where the assumption might hold less. However, an undirected mesostate transition map has been used. Whenever possible, the transition between two mesostates was routed through other mesostates, to avoid unnaturally wide movements of the protein chain, as shown in Figure 3.4. In the mesostate transition graph, the highest node degree helps determine the upper bound of the unfolding algorithm.

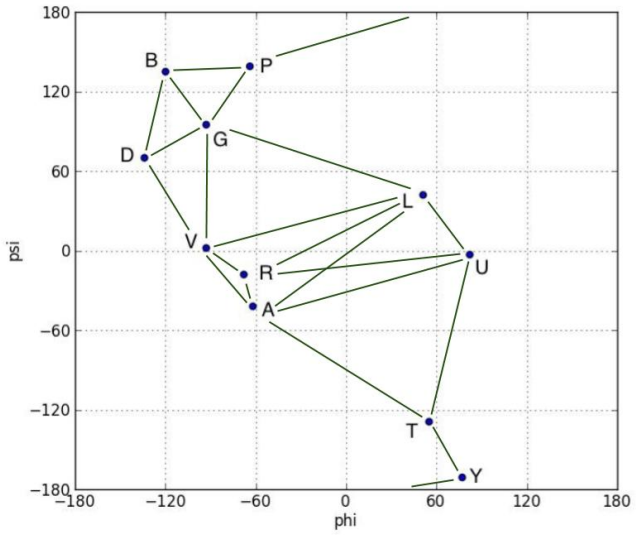

Figure 3.4: The undirected mesostate transition map, with a maximum node degree of 5.

3.2.5 Rigid and flexible amino acids

Another restriction of the search space that was implemented in this work was to, on a protein-by-protein basis, manually specify which amino acids are flexible. Only flexible amino acids can have their phi and psi values changed in the unfolding motion. This approach has been previously implemented (Fitzkee and Rose 2004; Minary and Levitt 2008). Initially, this selection of rigid segments was done automatically, when an amino acid was close to the mesostate of α-helices or β-sheets. Since there are mesostates that specifically encode these secondary structures, the selection of rigid amino acids is straightforward. However, by manually specifying the flexible residues, one can also select the parts of the sheets or helices whose unfolding could supposedly contribute to a faster unfolding speed.

There is one amino acid that also contributes to the rigidity of the main chain. Proline is an amino acid whose side chain fixes the phi angle because of an extra bond to the main chain. Therefore, the mesostate for proline is fixed in this implementation.

3.3 Scoring function

A scoring function has been used to guide the tree exploration. Similar to the energy function in protein motion planning algorithms (Al-Bluwi et al. 2012), the scoring function evaluates how likely it is for the protein to adopt a conformation. Unlike folding with motion planning, the potential energy associated with the current state of the protein cannot be calculated, because of the lack of side chains in the model. However, the present work does not rely on the main chain contributing to the scoring function, which has been argued for by Rose et al. (2006).

Two scoring functions have been implemented: smallest delta and radius of gyration. The smallest mesostate delta score is computed as the distance on the Ramachandran plot between the current mesostate and the future mesostate. The radius of gyration is used to evaluate the compactness of a protein, as a set of N atoms of mass m. The radius of gyration is calculated in two steps. First,

$$\sum_{i=1}^{N} m_i(r_i - R_C) = 0$$

gives the coordinates of the centre of mass R_C. Then the radius of gyration is calculated as the square root of sum of the squared distances from each atom r_i to the protein's centre of mass R_C divided by the mass of the molecule M:

$$R_g^2 = \sum_{i=1}^{N} m_i(r_i - R_C)^2 \,/\, M$$

Using all non-hydrogen atoms of the amino acids increases the computation time, without a considerable increase in fidelity (Lobanov et al. 2008). Therefore, a single atom per amino acid was used in this work in the calculation of the radius of gyration. The mass factor m_i goes away and the mass of the molecule M is substituted by N:

$$\sum_{i=1}^{N}(r_i - R_C) = 0 \text{ and}$$

$$R_g^2 \cong \sum_{i=1}^{N}(r_i - R_C)^2 \,/\, N$$

For each protein that the unfolding algorithm was tested on, the adequate threshold radius of gyration was selected based on the class of protein it belongs to and the sequence length (Lobanov et al. 2008).

3.4 Collision detection

To obtain an unfolding motion that respects the protein motion rules, the main chain has to be checked for collisions with itself. This is performed by checking the distance between pairs of atoms in 3D space against the sum of their radii. In this

context, Chimera uses the Van der Waals radius definition, which represents atoms as hard spheres.

In this work, the built-in Chimera extension for collision detection has been used. In the source code of Chimera, it is named `DetectClash` and references a `BOXES_METHOD` from binary file `closepoints.so`, for which the source code is not made available. Presumably, it implements the idea of dividing the 3D space occupied by the protein into boxes, as described by Levinthal (1966).

The box size guarantees both the collision threshold and the maximum number of atoms that can fit in a box. The box an atom belongs to is calculated from the atom's coordinates in constant time. On one hand, the query time for finding which box holds which atoms, or the reverse, is unitary. On the other hand, the boxing algorithm has to iterate through half of the atoms in the structure to test for collisions.

When performing collision detection, not all pairs of atoms need to be considered. Given an adequate box edge length, boxes can be used such that pairs are created only between atoms of neighbouring boxes: if two boxes do not share a side, an edge, or a corner, then the minimum distance between atoms in these boxes is larger than the collision threshold.

Potential movement-generated clashes need to also be avoided. Even if the start and end conformations are clash-free, a clash might occur during the transition between the two. In this work, if a motion incurs a radius of gyration difference greater than one, ten step-wise intermediate conformations are verified. However, if clashes are visible in the unfolding animation, the number of intermediate conformations to be verified has to be raised. The number of intermediate conformations also depends on the mesostate alphabet in use: the protein motion is more ample when the alphabet is reduced. Therefore, when using an expanded mesostate alphabet, fewer step-wise conformations need to be verified for clashes.

3.5 Graph exploration

Motion planning algorithms use an underlying graph data structure. In the context of this work, the nodes encode the protein conformation as a string of mesostates, one per amino acid. The directed edges represent the transition from a valid conformation to another. The edges do not store any information; they could have stored information regarding the mesostate that changes from the parent conformation to the child conformation. A protein chain movement is extracted from a starting node, a directed edge and an end node. The start and end nodes each store the string of mesostates encoding the main chain structure for the respective conformation.

During the graph exploration, all conformations tested for collisions are added as nodes in the graph. If a conformation has collisions, or has been already explored, it is avoided by the beam search. The edge connecting a parent node with a child node is added only if both the child node and the transition are clash free.

Because of the need of finding a realistic unfolding pathway, the graph has to be explored breadth-first. The branch and bound algorithm provides a hint of reducing

the search space by removing the poor solutions. In this work, a variation of branch and bound called beam search was used. Beam search (Reddy 1977) explores a tree breadth-first, but limits the exploration of the new level of the tree to a fixed number of nodes, as shown in Figure 3.5. Beam search can be seen an extension of an expanded best-first search (Pearl 1984), where the next iteration of the breadth-first search is seeded by a set of top scoring nodes, instead of just the top scoring node.

In motion planning algorithms the graph starts expanding from the start node, resembling a tree. In PRM and RRT, the graph is randomly expanded until the start and end nodes become connected. In this work, randomness is replaced an incremental exhaustive approach, in the sense that at each iteration all possible solutions are considered. All but the top scoring partial solutions are discarded according to the size of the beam before advancing to the next iteration.

The size and growth rate of the graph directly affect the overall algorithmic complexity. Beam search can reduce the complexity by limiting the number of nodes added to the graph. For each flexible amino acid there are up to five mesostate transitions m_t possible at any time, as previously shown in the mesostate transition map in Figure 3.4. Even when using the simplifications of the protein model, without a bounding algorithm the graph would exponentially by up to m_t nodes for each node. Given a number of flexible residues N_f, the graph can grow iteratively by up to $m_t N_f, (m_t N_f)^2, (m_t N_f)^3, \ldots, (m_t N_f)^i$ new nodes with breadth-first search. With a beam search of width b_w the graph can grow only by $b_w m_t N_f$ new nodes at each iteration.

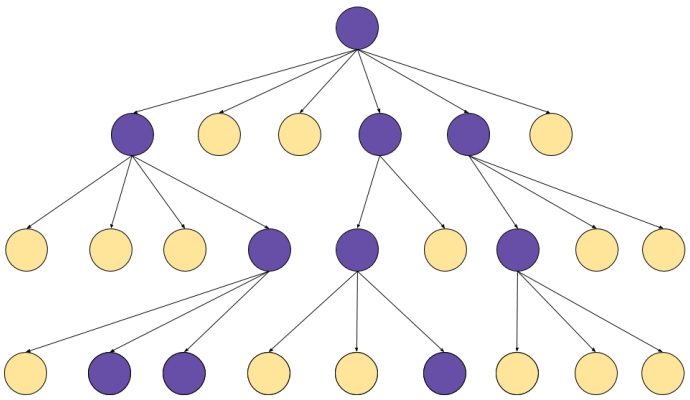

Figure 3.5: Example of beam search with size 3. On each layer, only 3 nodes in purple are explored further.

One disadvantage of beam search compared to an exhaustive search is that beam search does not guarantee the optimal solution. However, if it is presumed that the protein follows an approximate monotone function, the graph exploration does not require such a guarantee. Results show that coming up with a monotone scoring function might not be straightforward, but it is possible. In this work, the beam search exploration stops when a threshold value of the scoring function is reached.

4

Implementation

This work is based on an existing molecular visualisation software and on a graph library that facilitates motion planning.

4.1 Visualisation – Chimera

UCSF Chimera[1] 1.11.2 (Pettersen et al. 2004) – "an extensible visualization system" – was used in this work because it is a popular molecular visualisation software that also permits the development of extensions. By implementing the current work as an extension, a higher degree of user-friendliness was ensured.

When implementing the solution, one objective was high modularity. In this work, any functional module, for example mesostate definition and assignment, or tree exploration, could be exchanged independently. Chimera also supports cross-extension calls, which meant existing functionality could be used, instead of being implemented. Two Chimera extension have been used in this work: `Find Clashes / Contacts` to detect collisions and `Morph Conformations` to animate the protein chain through conformations extracted from the shortest pathway in the motion graph. With the help of other Chimera extensions, videos of the animation can be recorded.

Even though the development setup is difficult for the current version of Chimera, it is expected that it will be simplified for the currently unreleased UCSF ChimeraX. ChimeraX offers a one-click plugin installation from a central repository, handles plugin dependency and supports immersive visualisations through virtual reality. These features would be useful for a future version of the current work.

4.2 Motion planning – `graph-tool`

Motion planning algorithms use a graph data structure. Although this work explores the conformation space guided by beam search instead of randomness, it still relies on a graph data structure.

In this work, the `graph-tool` Python library was favoured over an independent solution, in order to simplify the installation procedure on another machine. Alternatively, a graph database could have been used, e.g. Neo4j (*Neo4j* n.d.). However,

[1] Chimera is developed by the Resource for Biocomputing, Visualization, and Informatics at the University of California, San Francisco (supported by NIGMS P41-GM103311).

such a setup would have been more difficult to replicate on other machines. Additionally, it would not have any desired benefit over `graph-tool`.

Moreover, `graph-tool` has implementations of search algorithms that are needed to extract the folding pathway from the explored tree. In this work, Dijkstra's algorithm was used to retrieve the path of conformations from the native conformation to the designated end conformation. The implementation of Dijkstra's algorithm has a lower time complexity than $A^* : O(E + V \log V)$ versus $O(E + V) \log V$, according to the graph-tool documentation.

Another advantage of `graph-tool` is that it has extended graph visualisation functionality, for example a function to export an image of the graph structure. Such images have been used in this work to track the progress of the otherwise opaque unfolding algorithm. A disadvantage of having `graph-tool` as a dependency is that it limits the portability of the extension to other machines. Also, it might be less performant than a dedicated implementation.

4.3 Development setup

For ease of distribution, Chimera comes packaged with its own Python environment, version 2.7.10. This version was the same as on the development machine, which facilitated the installation of `graph-tool` and its dependencies. Also, the Python package `numpy` within Chimera was incompatible with `graph-tool`; a symbolic link to a newer version was used in the Chimera installation folder. The installation validation was done in Chimera's Interactive DeveLopment Environment (IDLE), by appending the relevant system paths and outputting the package versions. Regular output in IDLE was noticed to be particularly slow. Another disadvantage of Chimera is that it runs on a single thread.

A symbolic link to the extension implemented in this work was placed in the installation path of Chimera, next to other extension that are packaged with Chimera. To load the recent code changes, the Python compiled files were deleted before reloading Chimera.

5

The `Unfolding` extension

The aim of this work was to obtain a user friendly way to generate and visualise the unfolding pathway of a generic protein. The user interface of the implemented extension is shown in Figure 5.1.

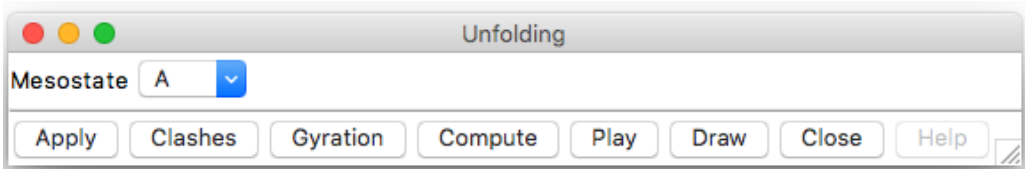

Figure 5.1: The `Unfolding` extension implemented in Chimera.

To use the extension, the desired protein structures can be fetched from the Protein Data Bank (Berman et al. 2000) by Chimera. The structure has to be edited according to the chosen one-bead model. Only the main chain atoms should be kept; the rest should be removed by using functions available in Chimera. Bonds between sulphur atoms that are formed between different amino acids restrict the movement of the main chain. By removing the side chains, these bonds are removed. If such bonds are present in the structure, the `Unfolding` extension will remove them automatically.

The extension implements several functions that are used on the residues manually selected by the user through Chimera's interface. The first button applies the selected mesostate to the selected residues. The `Clashes` button outputs to IDLE the number of atoms that create a steric clash, while `Gyration` outputs the radius of gyration.

The other buttons in the interface use the entire protein model, regardless of the user selection. Pressing the `Compute` button starts the calculations of the unfolding pathway. When the calculation is finished, the basic motion is given by the conformations that make the shortest path in the graph. For a smoother motion, a number of linear steps are computed for the final motion. Pressing the `Play` button advances the protein unfolding through the linear stepts of the final motion. The `Draw` button can be pressed at any time during the computation to generate a picture of the current state of the graph.

The extension can be used on any protein structure, but it assumes an edited structure according to the protein model and a single protein model opened in Chimera. For any chosen protein, there are a number of parameters that need to be configured before pressing `Compute`. In the current version of the extension, parameters such as the flexible amino acid indices, the target radius of gyration at which the unfolding

stops and the beam size are not visible in the GUI; they need to be set in code, before running Chimera.

To obtain an automatic animation, intermediate conformations need to be copied as distinct models and animated with the Morph extension in Chimera. After this step, the user has full control over the animation speed and direction in both folding and unfolding directions, in addition to normal Chimera options, such as molecular representation and 3D movement. Moreover, the Chimera scene can be saved and shared with others.

The Unfolding extension runs, like Chimera, in a single-threaded mode. Multi-threading is supported in Chimera; however, it is not part of the Chimera framework. Moreover, in order for the Find Clashes/Contacts extension to work, the protein must be in the desired conformation. One solution could be to duplicate the protein structure such that there are essentially more identical protein models to work with. Alternatively, a multi-threaded implementation would have to solve the collision detection problem in a different way than with Chimera's extension. However, given the generally short running time of the implemented extension, it might be that the difficulty of implementing a multi-threaded solution might, overall, yield insufficient benefits.

6

Results

This work set off to create animations for two proteins at opposite ends of the expected unfolding difficulty. Out of the two, only one animation was successfully created. This section will describe how the animation was created and continues by explaining potential reasons that hindered the obtaining of an animation for the other protein. To further evaluate the limitations of the `Unfolding` extension, unfolding a set of proteins of diverse typology was attempted. The mixed results are described in the last part of this section.

6.1 Studied proteins

Two proteins have been chosen as the focus of this work. They are small, compact and around 80 amino acids long. They present both helices and sheets, padded by coil segments that make a good choice for being marked as the flexible regions of the protein. These two proteins were chosen because they constitute a very easy and a very difficult case of unfolding, respectively.

Ubiquitin, in particular PDB entry 1UBQ, was chosen because it is included in a motion planning review (Al-Bluwi et al. 2012). However, the review does not present the video or animation that would show the protein motion. Moreover, the folding of ubiquitin has been computed with molecular dynamics simulations (Alonso and Daggett 1998); even though the authors have shared only images of the intermediate folding states, it constitutes as a basis for comparison. The structure of ubiquitin is made of two parts that can separate in 3D space without overlapping, which makes it an easy protein to unfold.

The other protein is the smallest knotted protein known, PDB entry 2EFV. Knotted proteins are of particular interest when studying protein mechanics because, even though they must be governed by the same rules as other proteins; "knots in the polypeptide chain were postulated as highly improbable due to a large entropic barrier to folding and the intrinsically difficult process of formation of knotted topology" (Andreeva 2016). Currently, only about 1% of the proteins in the PDB are knotted (Jackson et al. 2017). 2EFV has a shallow knot – by removing only a small number of amino acids from one end of the protein, the knot would disappear. This makes the protein an easier unfolding target than other proteins with deeper knots. Several recent articles have focused on the folding pathway of this protein (Sulkowska et al. 2012; Beccara et al. 2013; Wang et al. 2015). However, none have shared an animation or video of the unfolding process.

6.2 Unfolding 1UBQ

An unfolding pathway of ubiquitin was obtained with a set of parameters after just a few minutes of computation. The beam size was 6 and the scoring function used was the radius of gyration. The native conformation has a radius of gyration of 11. The computation stops after a few minutes, when the unfolded ubiquitin reaches a radius of gyration greater than 23, which becomes the final conformation of the unfolding. The intermediate mesostates were not needed for the unfolding of ubiquitin; therefore, the animation does not use them. However, the mesostate transition map was used. The flexible residues for 1UBQ are shown in Figure 6.1.

Figure 6.1: The flexible amino acids for 1UBQ highlighed in red, indexing from position 0: 6-9, 17-21, 34-38, 44-47, 50-54, 59-63. Only about 37% of the protein's chain is flexible.

The step-wise animation generated by the `Unfolding` extension is shown in Figure 6.4. These intermediates are visually similar to the reverse of those obtained by folding ubiquitin with molecular dynamics (Alonso and Daggett 1998). While a more accurate method of comparison is desirable, side-by-side image comparison is the only way to compare the two motions, because no video or animation was produced by Alonso and Daggett.

The extension also creates a series of images that show the beam search graph exploration as it progresses (Figure 6.2). Each node in the graph is labelled with its graph id and the associated mesostate transition, in the format of the old mesostate letter, the amino acid index, and the new mesostate letter. The nodes that are explored in each stage of the beam search are highlighted in yellow. According to the beam size value, the top scoring 6 nodes are selected to be explored in the next stage; these nodes are coloured differently. The graph is laid out like a radial tree. The start node is positioned at the centre and each beam search stage has a different radius. Figure 6.3 shows how the Dijkstra's algorithm retrieves the shortest path from final graph.

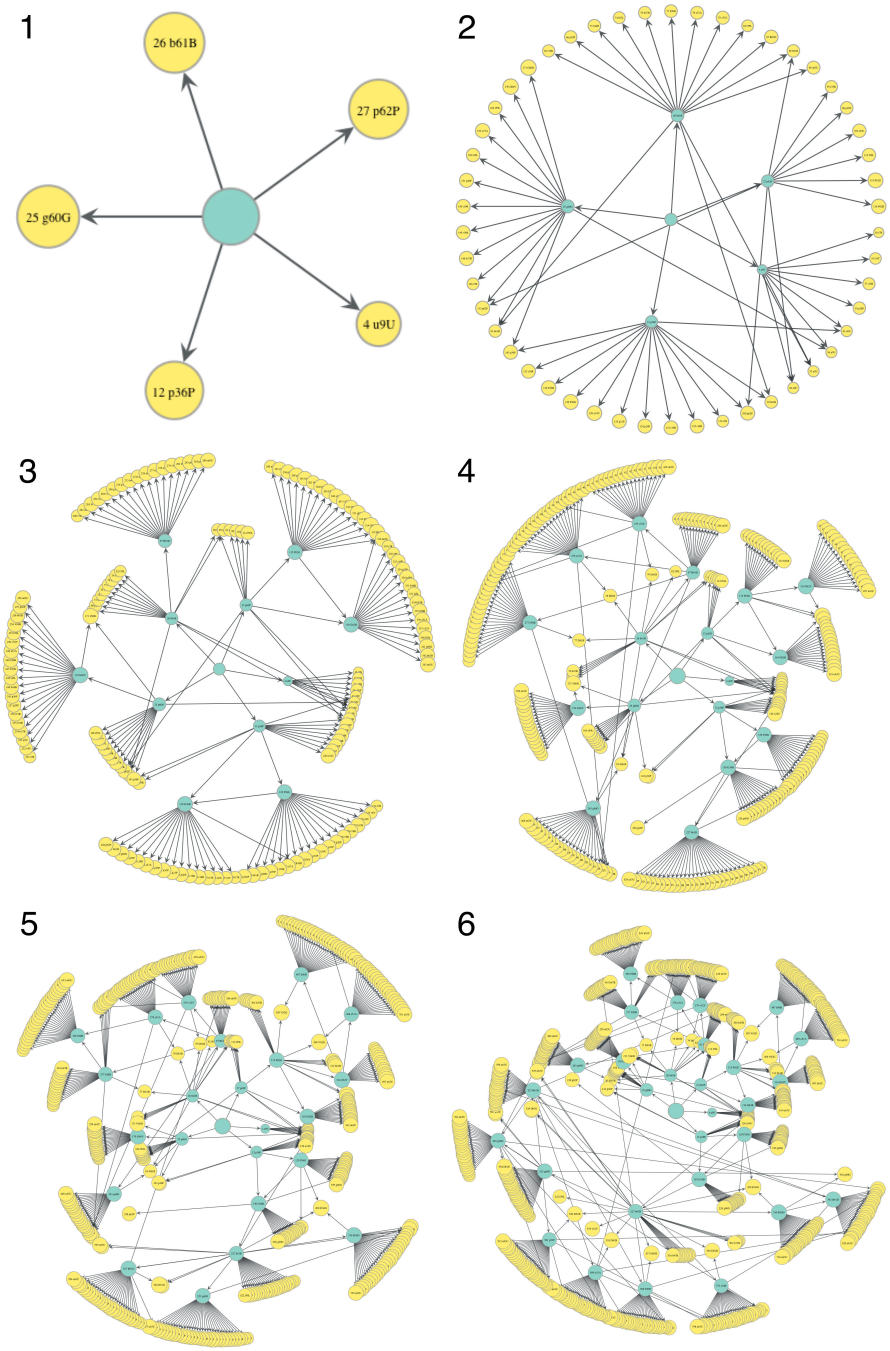

Figure 6.2: The expansion of the unfolding graph for ubiquitin, in a radial tree layout. The explored nodes are highlighted in cyan. The expanding radii correspond to the tree levels, as created by the beam search exploration. In step 1, only five mesostate changes can be made from the starting native conformation: mesostate G for amino acid 60, B for 61, P for 62, U for 9 and P for 36. The small number of possible mesostates is because the phi and psi angles first need to be approximated to a mesostate before transitions to other mesostates can be made. The small letters on the labels indicate that the phi and psi angles at those positions have the original values, and to which mesostate they are closest to. In step 2, all 5 mesostates are accepted by the beam and are consequently explored. The top scoring 6 mesostates are further explored in step 3 and so on.

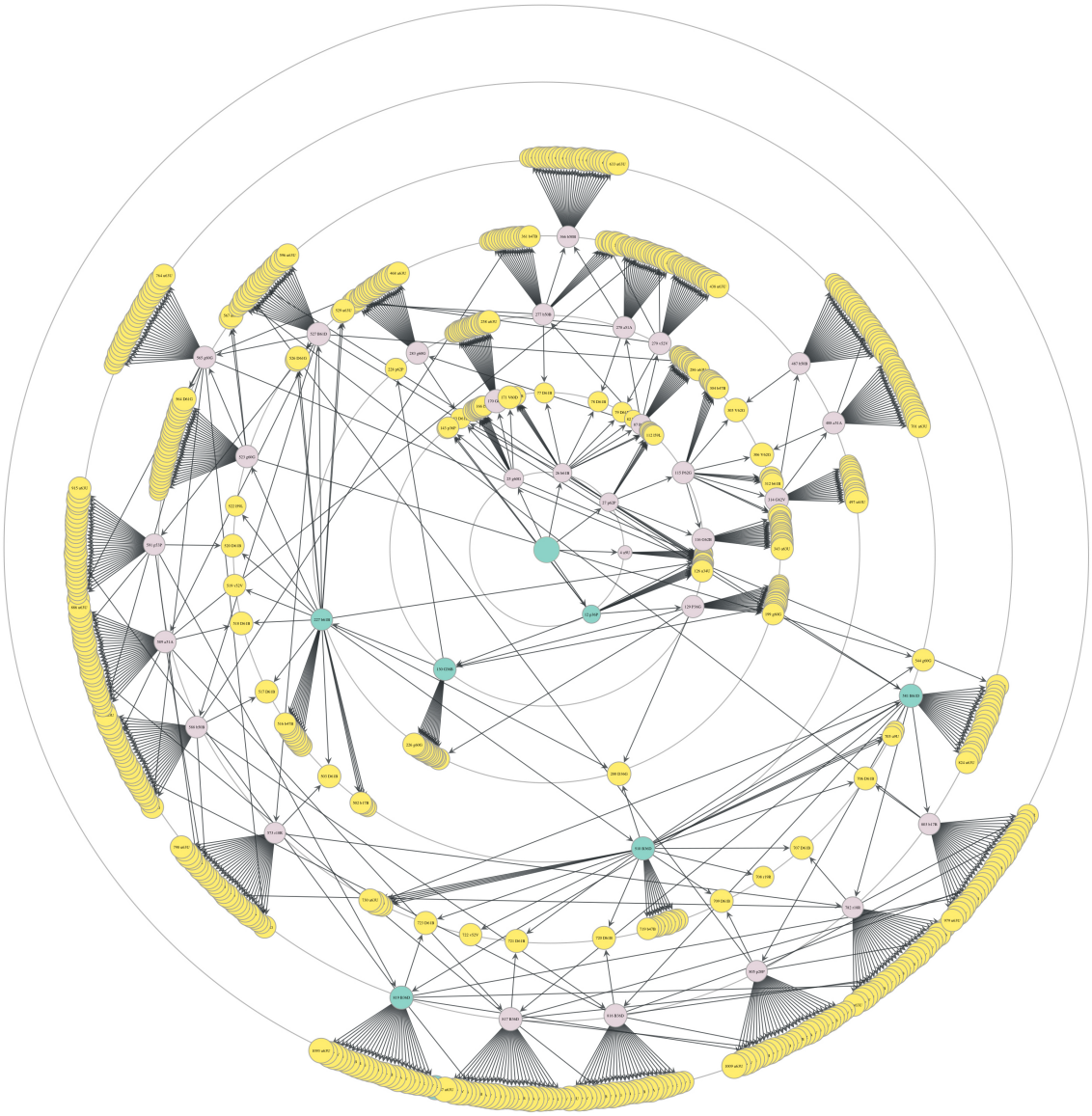

Figure 6.3: The final graph for 1UBQ. The circles highlight the tree exploration iterations, with the start node in the centre. All nodes are clash-free mesostate transitions. The nodes that are discarded at each iteration are in yellow. The nodes in cyan give the unfolding motion; they are retrieved by Dijkstra's algorithm.

6.3 Unfolding 2EFV

When unfolding the knotted protein, the same parameters were used as for ubiquitin. It was quickly observed that the mesostate transitions were too large and generated collisions. Therefore, the intermediate mesostates were added to the transition map.

Both scoring functions have been tested. The function preferring the smallest moves applied multiple protein motions without clashes. However, since this function does not differentiate between a larger and a smaller radius of gyration, the protein was

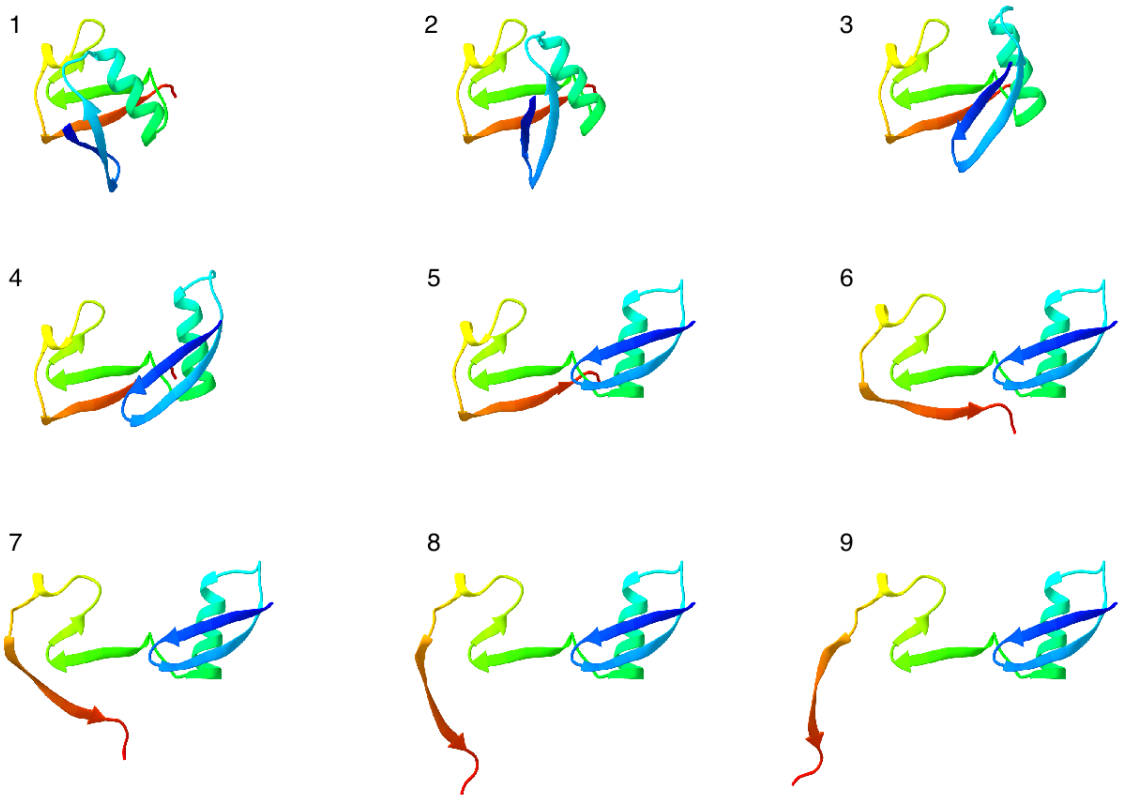

Figure 6.4: The unfolding animation of 1UBQ obtained with the `Unfolding` extension implemented in Chimera.

not guided towards and unfolded state. With a strict hierarchy, the protein was guided towards unfolding; however, by exhausting the smallest moves quickly, the protein lost the initial flexibility and the knot was not unfolded. The other function of maximising the radius of gyration prefers large moves, which led the protein to trapped conformations, where it could not increase the radius of gyration without generating clashes.

The running time and graph size were much larger than for ubiquitin, despite similar protein sizes. Therefore, new scoring functions should be tested when unfolding knotted proteins.

6.4 Further evaluation

The `Unfolding` extension was tested on a small set of proteins selected by Plaxco et al. in Table 1 (1998). These 12 non-homologous proteins have diverse topologies and are 67 to 104 amino acids long. They are listed in Table 6.1, together with the results of the unfolding.

The unfolding of PDB entry 1UBQ was already described in a previous subsection. PDB entry 1PCA was skipped because its identifier does not match the description of the protein in the table.

PDB entry	Topology	Chain length	Unfolding successful
1LMB	helical	80	yes
1HRC	helical	104	yes
2ABD	helical	86	yes
1UBQ	mixed	76	yes
1CIS	mixed	83	no
1PCA	mixed	80	skipped
2PTL	mixed	63	yes
1HDN	mixed	85	no
1APS	mixed	98	not quite
1CSP	sheet	67	no
1TEN	sheet	90	no
1SHF	sheet	67	yes*

Table 6.1: Evaluation of the `Unfolding` extension of 12 topologically diverse proteins. The success rate for this set is approximately 50%, which is not surprising given the very reduced conformational space.

PDB entries 1LMB, 1HRC, 2ABD and 2PTL have been unfolded successfully with the same parameters as 1UBQ. Some entries have long main chain regions which are neither α-helices nor β-sheets. Instead of marking every such amino acid as flexible, only every second or third was chosen. Images of the start and end conformations are included in Appendix A.

PDB entry 1SHF was unfolded using more relaxed parameters. The beam size was increased to 14, and intermediate mesostates were used. Moreover, all amino acids were marked as flexible. Given how β-sheets are formed, this assumption might be reasonable.

PDB entry 1APS struggled to unfold using the same relaxed parameters as 1SHF. Overall, the behaviour was similar to the unfolding of 2EFV, in the sense that some motion was observed, however only in one part of the protein; even when running the algorithm for a long time these proteins did not unfold. This suggests that maximising the radius of gyration is an inadequate scoring function for unfolding of all proteins.

The remaining PDB entries 1CIS, 1HDN, 1CSP and 1TEN did not unfold at all, regardless of how relaxed the parameters were. For these four proteins, at most one amino acid could be approximated to a mesostate without generating clashes. Once the amino acid used a mesostate, no further mesostate transitions were possible. Other times, no amino acid could use a mesostate. One could say that for these four proteins the unfolding was not able to start.

One factor that might contribute to the limited success of the current implementation is that one amino acid, proline, rigidifies the main chain in a way that the proline's mesostate cannot be changed.

Another factor is that the loop regions between α-helices are typically longer than the ones connecting β-sheets. This implies that the mesostate transitions for structures

rich in β-sheets need to be finer and that more mesostates are required to unfold such structures.

In some cases, the protein structure is interlocking – without applying two mesostate transitions in parallel, clashes are generated. Overall, the very reduced conformational space used in this work is incompatible with some protein structures. Even the expansion of the mesostate alphabet as intermediate mesostates was shown to be, in some cases, insufficient.

7

Discussion

Attempting to visualise protein movements is an old idea. A one-dimensional protein chain was folded into a 3D shape with an algorithm that divides the folded structure into a tree of smaller components (Tsai et al. 2000). However, the generated visualisation is not animated nor interactive. Amato and Song (2002) have visualised the folding pathways of several short proteins with motion planning. Again, the visualisations were not animated nor integrated with molecular visualisation software. Fitzkee and Rose (2004) have used a rigid segment model to statistically analyse clash-free states. However, these states were not used to construct a protein chain animation.

This work has implemented an extension of a molecular visualisation software that successfully unfolds some proteins in a very restricted conformational space. In order for the extension to work reliably on various classes of proteins, further improvements should be made to the protein model, the applied algorithms and the scoring functions.

Helices and sheets have been kept intact in the unfolding, since they are arguably stable structures. One could, however, permit the unfolding of these secondary structures in a hierarchical order, starting from the ends, since the ends are structurally more flexible than the rest.

A two-bead protein model could be used in this work. Although the atomic model of the protein was reduced to the main chain through the complete removal of all side chains, the side chains could be abstracted to a single virtual atom (Levitt and Warshel 1975). The presence of these virtual atoms would mimic the position of the side chain. Additionally, the radius of a virtual atom can be adjusted to reflect the volume of the side chain, thus generating more clashes that resembles more closely that of the full protein model. Moreover, this would not have had a significant impact on the overall speed of the computation.

7.1 Mesostates

There current work relies on a static mesostate alphabet. It might be possible to create an alphabet based on the protein to be unfolded. This approach could also take into account the special cases of specific amino acids - the trans conformation of prolines, and the flexible glycines which have very limited motion in the 11 mesostate alphabet.

Alternatively, one could create a mesostate alphabet tailored to the type of amino

acid and to the secondary structures connected by the respective amino acid. Any loop connecting β-sheets needs more degrees of freedom than if it were to connect α-helices, as shown by the lack of success in unfolding β-sheet structures.

The current work assumes a linear transition between mesostates – the lines in Figure 3.4. The transition between mesostates is, in reality, probably non-linear. Perhaps a slightly curved transition would avoid the clashes that appear during a linear transition. A solution could be to randomly test various phi and psi variations close to the linear transition, especially if the number of detected clashes is low, and if they occur only in a limited portion of the movement between the mesostates. Another approach would be to use a different kind of collision detection algorithm. For example, the Grid algorithm observes the space between the atom spheres (Moult and James 1986).

A limitation of the current work pertains to the mesostates defined by Chellapa and Rose (2012). These mesostates are squares of 20°. A residue can have many variations of phi and psi values, while still being within the same mesostate. In this work, mesostates are conceived as pairs of a single phi value and a single psi value, whereas they were originally defined as all possible phi and psi each within 10 degrees of the base values. This flexibility has not been used in the implemented algorithms.

It is perhaps the definition of a mesostate that limits the potential of the `Unfolding` extension. When transitioning between two mesostates, both the phi and psi bonds need to be rotated. If a transition would be applied one bond at a time, the chain motion would look different and the protein might be able to unfold.

7.2 Scoring function

The scoring function used in this work has the role of measuring the unfolding degree of proteins. Such a function has not been identified by researchers. However, existing folding functions could be reversed and tested for their unfolding potential. A new kind of scoring function could be inspired by extracting coarse-grained potentials for local interactions in unfolded proteins (Ghavami et al. 2013).

The unfolding of ubiquitin was obtained by maximising the radius of gyration. Because of the calculus formula of this radius, this scoring function prioritises moving apart big parts of the chain. If one considers the mesostate deltas to be the same for all transitions, the more atoms are moved away from the centre of the protein, the more the radius of gyration increases. This effect is arguably a reverse of the hydrophobic collapse, a protein folding theory suggested by Levitt and Warshel (1975 as cited by Karplus and Weaver 1994). This theory has been verified by computer simulations on ubiquitin and is line with experiments (Alonso and Daggett 1998).

The graph of the radius of gyration obtained from molecular dynamics is not monotonic (Figure 12.A in Alonso and Daggett 1998). If the graph is observed in reverse, in accordance to an unfolding process, the radius of gyration suddenly decreases. This suggests that the scoring function based on the radius of gyration is not suitable as a scoring function for the entire unfolding process.

Instead of testing scoring functions which are based on distinct principles, several functions can be combined. For example, the radius of gyration and the mesostate delta could be combined in a single function. Moreover, each contributing function could be associated a variable weight. In the beginning of the unfolding, one could allow only small variations in shape, followed by larger movements corresponding to a hydrophobic collapse, as suggested by Chan and Dill for a hen egg white protein (1998). Later in the unfolding, the radius of gyration could have a bigger weight, so as to speed up the unfolding.

7.3 Collision detection

Since the collision detection algorithm runs at least once for every potential conformation, any small improvement could have a significant impact, which would be more noticeable the larger the protein.

An improvement would be to filter which pairs of atoms are verified for collisions. There are three types of clashes, depending on the atoms: between the moving and fixed atoms, between the moving atoms themselves, and between the fixed atoms themselves. Given that the molecule is checked for clashes initially and continuously as it is moved, there is no need to check the fixed atoms against themselves at every mesostate transition.

The current work verifies in-motion clashes, when the protein chain moves from one conformation to another, in a fixed number of increments. However, the number of intermediate collision checks should be linked with the amplitude of the motion, and the length of the portion of the chain that moves. In motion, Chimera keeps the longer part of the chain fixed and rotates the shorter one. The furthest atoms of the chain travel a distance proportional to the distance to the centre of movement.

Since collision detection checks need to be run every time an atom moves more than a certain threshold, the number of intermediate conformations has to be proportional with both the mesostate delta and the distance to the furthest atom in motion. If the number of intermediate conformations is too small, the collision detection would fail by allowing collisions for the conformations that were not checked, leading to an impossible protein movement. The current work verifies a static number of in-motion conformations to be clash free. It would be more suitable to dynamically compute how many in-motion conformations need to be verified.

A more recent algorithm, called BioCD (Ruiz de Angulo et al. 2005), hints at a potential improvement over Chimera's `BOXES_METHOD`. BioCD uses bounding volume hierarchies, a tree representation of geometric objects wrapped in bounding volumes. This approach would adapt well to the current method because it could bound the already rigid helices and sheets to volumes. Moreover, BioCD only compares atoms whose position was changed. Presumably, Chimera is not tracking which atoms have moved; therefore it is probably not benefiting from this improvement.

7.4 Graph exploration

This work facilitates research in two ways. First, the protein motion can be observed and interacted with in Chimera. Second, with the use of `graph-tool`, the graph can be monitored and analysed in parallel with the computation.

One particular case to be observed in the graph is nodes with more than one incoming edge, for example node 581 in Figure 6.3 which is part of the unfolding pathway. Most nodes in the graph have a single incoming edge. If a node has multiple incoming edges this might suggest that this particular transition is of interest, since the protein reaches the same conformation in different ways. This is in line with the idea of folding intermediates, as described by Radford et al. (1992).

More information about the unfolding pathway could be extracted if multiple motion planning graph are compared. By changing some of the key variables, the graph might differ. The nodes that persist across several graphs could indicate a more robust unfolding state.

7.4.1 Improvements

In this work, the unfolding graph has been explored with an implementation of beam search. Other approaches that could yield better results are described below.

One improvement could be to add memory to the beam. The beam search is conceptually a greedy approach, by only selecting the highest scoring solutions from the current iteration. However, it might happen that a solution from a previous stage is better than some or all solutions currently chosen according to the beam size. In this case, an even greedier approach is to always remember the top partial solutions, and compare the results of the current iteration against them. In contrast to beam stack search (Zhou and Hansen 2005) which guarantees completeness, there is no need of backtracking to previous solutions. The exploration algorithm could always remember the top scoring nodes and merge the new top scoring nodes, after which it applies the pruning according to the beam size.

Implementing memory over beam search could yield other improvements. If a mesostate change does not lead to a significant improvement of the scoring across several beam search stages, perhaps it is not significant. For example, if a certain conformation is consistently selected by the beam search to be explored in the next iteration, but does not provide any results to the following iteration, it could be disfavoured in the future.

Another way of improving the beam search would be to apply the beam size restriction after exploring two tree levels instead of a single one. However, the number of explored nodes will increase significantly.

Another potential improvement is to adjust the beam size while the tree exploration is running. This is called iterative weakening (Provost 1993 as cited by Zhou and Hansen 2005). When the protein starts unfolding, the scoring function might not be accurate enough to distinguish between the mesostate transitions, especially in the case of the radius of gyration. Therefore, in the initial stages the beam size could be

large. However, later on in the motion, the difference between the gyration radii of mesostate transitions widens. Thus, the beam size can be reduced in the later stages of the unfolding. Iterative weakening needs to be adjusted for the scoring function, and maybe even for the protein to be unfolded, in order to avoid the pruning of too many nodes too early in the graph exploration.

8

Conclusions

This work has shown that it is possible to visualise the unfolding pathway through a motion planning approach, and how to obtain it with a constrained conformational space. The animation is interactive and the user has an unprecedented level of control of the visualisation, thanks to the integration of the solution with a molecular visualisation software. Moreover, the animation can also be recorded as a video, similar to existing protein motion visualisations.

After combining several simplifications from previous research, the pathway takes minutes to compute in the same thread as the Chimera application for a number of proteins. The unfolding of ubiquitin concurs with the reverse of the folding motion obtain in other studies. Unfolding animations were obtained for several topologically different proteins. However, more work is needed to further refine the mesostates, the graph exploration methods and the scoring functions, as shown by the attempt to unfold a knotted protein and several other β-sheet rich proteins.

Overall, the work shows that it is possible to have a platform that reopens the debate about a high level function that governs the protein motion, a platform where the effects of these motion guiding functions can be seen.

Bibliography

Alonso, D. O. and V. Daggett (1998). "Molecular dynamics simulations of hydrophobic collapse of ubiquitin". In: *Protein Science* 7.4, pp. 860–874. DOI: 10.1002/pro.5560070404.

Amato, N. M. and G. Song (2002). "Using motion planning to study protein folding pathways". In: *Journal of Computational Biolology* 9.2, pp. 149–168.

Andreeva, A. (2016). "Lessons from making the Structural Classification of Proteins (SCOP) and their implications for protein structure modelling". In: *Biochemical Society Transactions* 44.3, pp. 937–943. DOI: 10.1042/BST20160053.

Baker, E. N. and R. E. Hubbard (1984). "Hydrogen bonding in globular proteins". In: *Progress in Biophysics and Molecular Biology* 44.2, pp. 97–179. DOI: 10.1016/0079-6107(84)90007-5.

Beccara, S. a et al. (2013). "Folding Pathways of a Knotted Protein with a Realistic Atomistic Force Field". In: *PLoS Computational Biology* 9.3. DOI: 10.1371/journal.pcbi.1003002.

Berman, H. M. et al. (2000). "The Protein Data Bank". In: *Nucleic Acids Research* 28.1, pp. 235–242.

Al-Bluwi, I., T. Siméon, and J. Cortés (2012). "Motion planning algorithms for molecular simulations: A survey". In: *Computer Science Review* 6.4, pp. 125–143. DOI: 10.1016/j.cosrev.2012.07.002.

Chan, H. S. and K. A. Dill (1998). "Protein folding in the landscape perspective: chevron plots and non-Arrhenius kinetics". In: *Proteins* 30.June 1997, pp. 2–33.

Chellapa, G. D. and G. D. Rose (2012). "Reducing the dimensionality of the protein-folding search problem". In: *Protein Science* 21.8, pp. 1231–1240. DOI: 10.1002/pro.2106.

DasGupta, D., R. Kaushik, and B. Jayaram (2015). "From Ramachandran Maps to Tertiary Structures of Proteins". In: *Journal of Physical Chemistry B* 119.34, pp. 11136–11145. DOI: 10.1021/acs.jpcb.5b02999.

Dill, K. A. and J. L. MacCallum (2012). "The Protein-Folding Problem, 50 Years On". In: *Science* 338.November, pp. 1042–1047. DOI: 10.1126/science.1219021.

Dobson, C. M. (2003). "Protein folding and misfolding". In: *Nature* 426.December, pp. 884–890. DOI: 10.1038/nature02261.

Fersht, A. R. and V. Daggett (2002). "Protein Folding and Unfolding at Atomic Resolution". In: *Cell* 108.4, pp. 573–582. DOI: 10.1016/S0092-8674(02)00620-7.

Fitzkee, N. C. and G. D. Rose (2004). "Reassessing random-coil statistics in unfolded proteins". In: *PNAS* 101.34, pp. 12497–12502. DOI: 10.1073/pnas.0404236101.

Fitzkee, N. C. and G. D. Rose (2005). "Sterics and solvation winnow accessible conformational space for unfolded proteins". In: *Journal of Molecular Biology* 353, pp. 873–887. DOI: 10.1016/j.jmb.2005.08.062.

Freddolino, P. L. et al. (2010). "Challenges in protein-folding simulations". In: *Nature Physics* 6.10, pp. 751–758. DOI: 10.1038/nphys1713.

Ghavami, A., E. van der Giessen, and P. R. Onck (2013). "Coarse-grained potentials for local interactions in unfolde proteins". In: *J. Chem. Theory Comput.* 9, pp. 432–440. DOI: 10.1021/ct300684j.

Gong, H., P. J. Fleming, and G. D. Rose (2005). "Building native protein conformation from highly approximate backbone torsion angles". In: *PNAS* 102.45, pp. 16227–16232. DOI: 10.1073/pnas.0508415102.

Jackson, S. E., A. Suma, and C. Micheletti (2017). "How to fold intricately: using theory and experiments to unravel the properties of knotted proteins". In: *Current Opinion in Structural Biology* 42, pp. 6–14. DOI: 10.1016/j.sbi.2016.10.002.

Kabsch, W. and C. Sander (1984). "On the use of sequence homologies to predict protein structure: identical pentapeptides can have completely different conformations". In: *Proceedings of the National Academy of Sciences of the United States of America* 81.4, pp. 1075–1078. DOI: 10.1073/pnas.81.4.1075.

Kabsch, W. and C. Sander (1983). "Dictionary of protein secondary structure: Pattern recognition of hydrogen bonded and geometrical features". In: *Biopolymers* 22, pp. 2577–2637. DOI: 10.1002/bip.360221211.

Karplus, M., A. Caflisch, et al. (1995). "Protein dynamics: From the native to the unfolded state and back again". In: *Molecular Engineering* 5, pp. 55–70. DOI: 10.1007/BF00999578.

Karplus, M. and D. L. Weaver (1994). "Protein folding dynamics: the diffusion-collision model and experimental data". In: *Protein Science* 3, pp. 650–668. DOI: 10.1002/pro.5560030413.

Kubelka, J., J. Hofrichter, and W. A. Eaton (2004). "The protein folding 'speed limit'". In: *Current Opinion in Structural Biology* 14, pp. 76–88. DOI: 10.1016/j.sbi.2004.01.013.

Lau, K. F. and K. A. Dill (1989). "A lattice statistical mechanics model of the conformational and sequence spaces of proteins". In: *Macromolecules* 22.10, pp. 3986–3997. DOI: 10.1021/ma00200a030.

Levinthal, C. (1966). *Molecular Model-building by Computer*. DOI: 10.1038/scientificamerican0666-42.

Levitt, M. and A. Warshel (1975). "Computer simulation of protein folding". In: *Nature* 253.5494, pp. 694–698. DOI: 10.1038/253694a0.

Lobanov, M. I., N. S. Bogatyreva, and O. V. Galzitskaia (2008). "Radius of gyration is indicator of compactness of protein structure". In: *Molecular Biology* 42.4, pp. 623–628.

Mayor, U., N. R. Guydosh, et al. (2003). "The complete folding pathway of a protein from nanoseconds to microseconds". In: *Nature* 421, pp. 863–867. DOI: 10.1038/nature01428.

Mayor, U., C. M. Johnson, et al. (2000). "Protein folding and unfolding in microseconds to nanoseconds by experiment and simulation". In: *Pnas* 97.25, pp. 13518–13522. DOI: 10.1073/pnas.250473497.

McCammon, J. A. et al. (1980). "Helix-coil transitions in a simple polypeptide model". In: *Biopolymers* 19, pp. 2033–2045. DOI: 10.1002/bip.1980.360191108.

Minary, P. and M. Levitt (2008). "Probing Protein Fold Space with a Simplified Model". In: *Journal of Molecular Biology* 375, pp. 920–933. DOI: 10.1016/j.jmb.2007.10.087.

Moult, J. and M. N. G. James (1986). "An algorithm for determining the conformation of polypeptide segments in proteins by systematic search". In: *Proteins: Structure, Function, and Bioinformatics* 1.2, pp. 146–163. DOI: 10.1002/prot.340010207.

Myers, J. K. and T. G. Oas (2002). "Mechanisms of fast protein folding". In: *Annual review of biochemistry* 71.1, pp. 783–815.

Neo4j (n.d.). URL: https://neo4j.com.

Nobel Media AB (n.d.). *The Nobel Prize in Chemistry 1972*. URL: http://www.nobelprize.org/nobel_prizes/chemistry/laureates/1972.

Onuchic, J. N. et al. (1995). "Toward an outline of the topography of a realistic protein-folding funnel". In: *PNAS* 92, pp. 3626–3630. DOI: 10.1073/pnas.92.8.3626.

Park, B. H. and M. Levitt (1995). "The complexity and accuracy of discrete state models of protein structure". In: *Journal of molecular biology* 249.2, pp. 493–507. DOI: 10.1006/jmbi.1995.0311.

Pearl, J. (1984). "Heuristics: intelligent search strategies for computer problem solving". In:

Pettersen, E. F. et al. (2004). "UCSF Chimera - A visualization system for exploratory research and analysis". In: *Journal of Computational Chemistry* 25.13, pp. 1605–1612. DOI: 10.1002/jcc.20084.

Plaxco, K. W. and M. Gross (2001). "Unfolded, yes, but random? Never!" In: *Nat. Struc. Bio.* 8.8, pp. 659–660. DOI: 10.1038/90349.

Plaxco, K. W., K. T. Simons, and D. Baker (1998). "Contact order, transition state placement and the refolding rates of single domain proteins". In: *Journal of Molecular Biology* 277, pp. 985–994. DOI: 10.1006/jmbi.1998.1645.

Porter, L. L. et al. (2015). "Subdomain Interactions Foster the Design of Two Protein Pairs with ˜80% Sequence Identity but Different Folds". In: *Biophysical Journal* 108.1, pp. 154–162. DOI: 10.1016/j.bpj.2014.10.073.

Provost, F. J. (1993). "Iterative weakening: Optimal and near-optimal policies for the selection of search bias". In: *AAAI*, pp. 749–755.

Radford, S. E., C. M. Dobson, and P. A. Evans (1992). "The folding of hen lysozyme involves partially structured intermediates and multiple pathways". In: *Nature* 358.6384, pp. 302–307. DOI: 10.1038/358302a0.

Reddy, D. R. (1977). *Speech Understanding Systems: A Summary of Results of the Five-Year Research Effort*.

Rose, G. D. et al. (2006). "A backbone-based theory of protein folding". In: *PNAS* 103.45, pp. 16623–33. DOI: 10.1073/pnas.0606843103.

Ruiz de Angulo, V., J. Cortés, and T. Siméon (2005). "BioCD: An efficient algorithm for self-collision and distance computation between highly articulated molecular models". In: *Robotics: Science and Systems*, pp. 241–248.

Samudrala, R. et al. (2000). "Constructing side chains on near-native main chains for ab initio protein structure prediction". In: *Protein Engineering* 13.7, pp. 453–457.

Srinivasan, R. and G. D. Rose (1995). "LINUS: a hierarchic procedure to predict the fold of a protein". In: *Proteins* 22.2, pp. 81–99.

Sulkowska, J. I., J. K. Noel, and J. N. Onuchic (2012). "Energy landscape of knotted protein folding". In: *Proceedings of the National Academy of Sciences* 109.44, pp. 17783–17788. DOI: 10.1073/pnas.1201804109.

Taylor, W. R. (2000). "A deeply knotted protein structure and how it might fold". In: *Nature* 406.August, pp. 916–919. DOI: 10.1038/35022623.

Taylor, W. R. (2007). "Protein knots and fold complexity: Some new twists". In: *Computational Biology and Chemistry* 31.3, pp. 151–162. DOI: 10.1016/j.compbiolchem.2007.03.002.

Tozzini, V. (2005). "Coarse-grained models for proteins". In: *Current Opinion in Structural Biology* 15, pp. 144–150. DOI: 10.1016/j.sbi.2005.02.005.

Tsai, C.-J., J. V. Maizel, and R. Nussinov (2000). "Anatomy of protein structures: visualizing how a one-dimensional protein chain folds into a three-dimensional shape". In: *PNAS* 97.22, pp. 12038–12043. DOI: 10.1073/pnas.97.22.12038.

Wang, I., S. Y. Chen, and S. T. D. Hsu (2015). "Unraveling the folding mechanism of the smallest knotted protein, MJ0366". In: *Journal of Physical Chemistry B* 119, pp. 4359–4370. DOI: 10.1021/jp511029s.

Ward, J. J. et al. (2004). "Prediction and Functional Analysis of Native Disorder in Proteins from the Three Kingdoms of Life". In: *Journal of Molecular Biology* 337.3, pp. 635–645. DOI: 10.1016/j.jmb.2004.02.002.

Wright, P. E. and H. J. Dyson (1999). "Intrinsically unstructured proteins: re-assessing the protein structure-function paradigm". In: *Journal of Molecular Biology* 293.2, pp. 321–331. DOI: 10.1006/jmbi.1999.3110.

Zhou, R. and E. A. Hansen (2005). "Beam-Stack Search: Integrating Backtracking with Beam Search". In: *ICAPS*, pp. 90–98.

A

Appendix 1

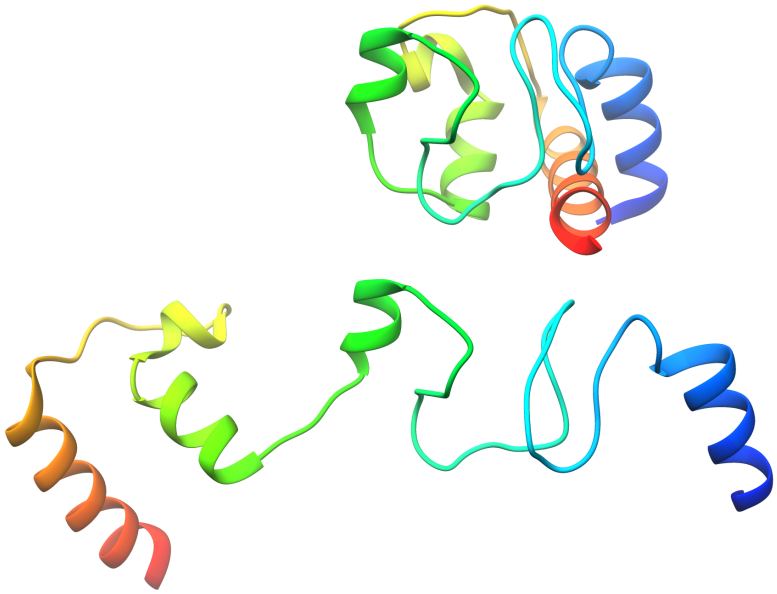

Figure A.1: Start and end conformations of the unfolding animation of 1HRC.

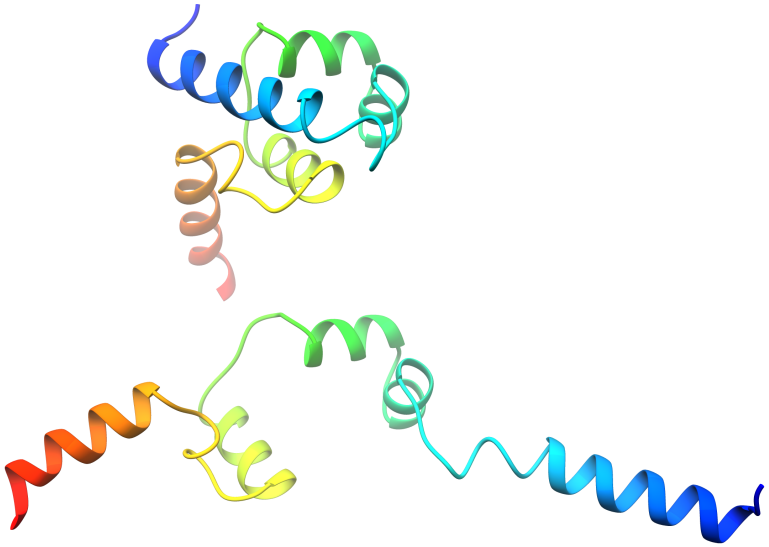

Figure A.2: Start and end conformations of the unfolding animation of 1LMB.

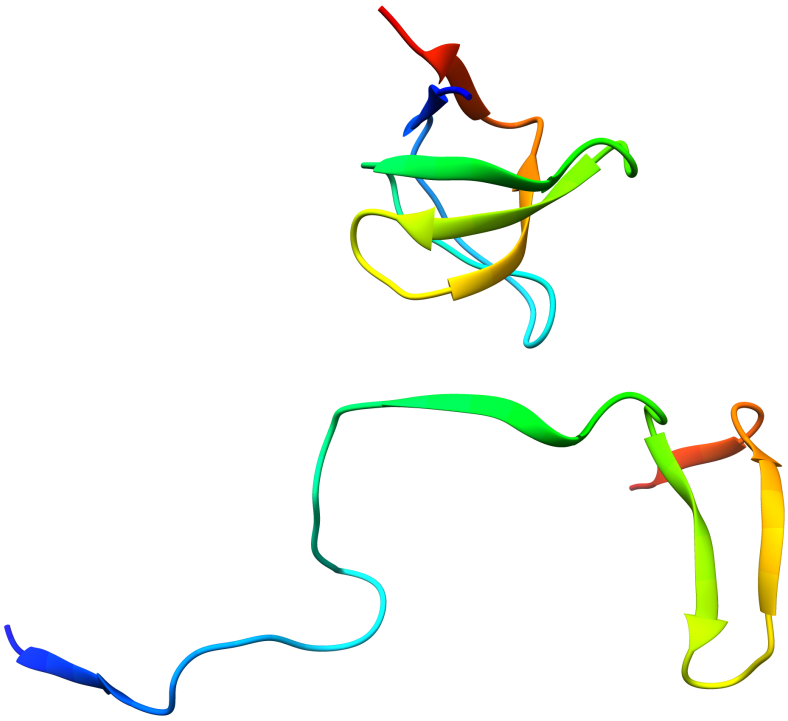

Figure A.3: Start and end conformations of the unfolding animation of 1SHF.

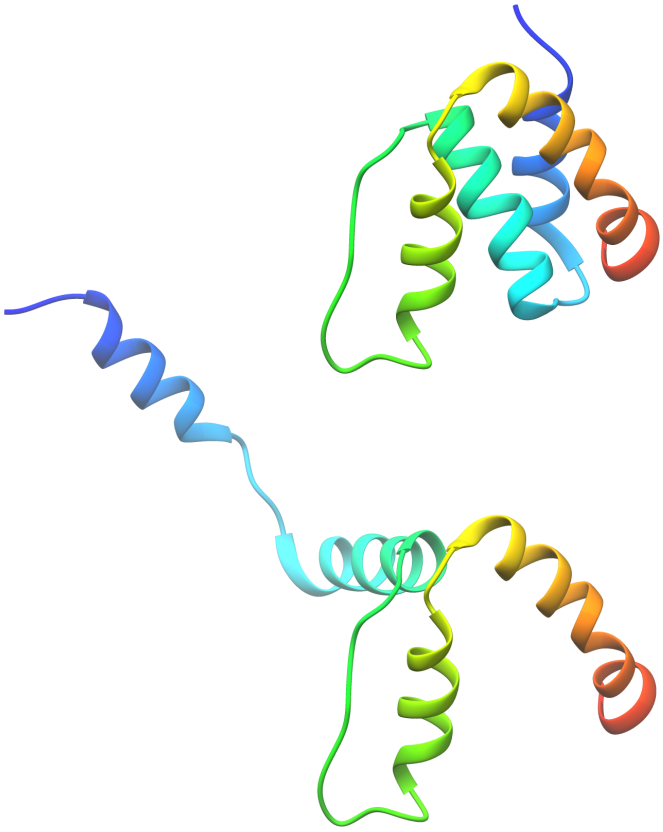

Figure A.4: Start and end conformations of the unfolding animation of 2ABD.

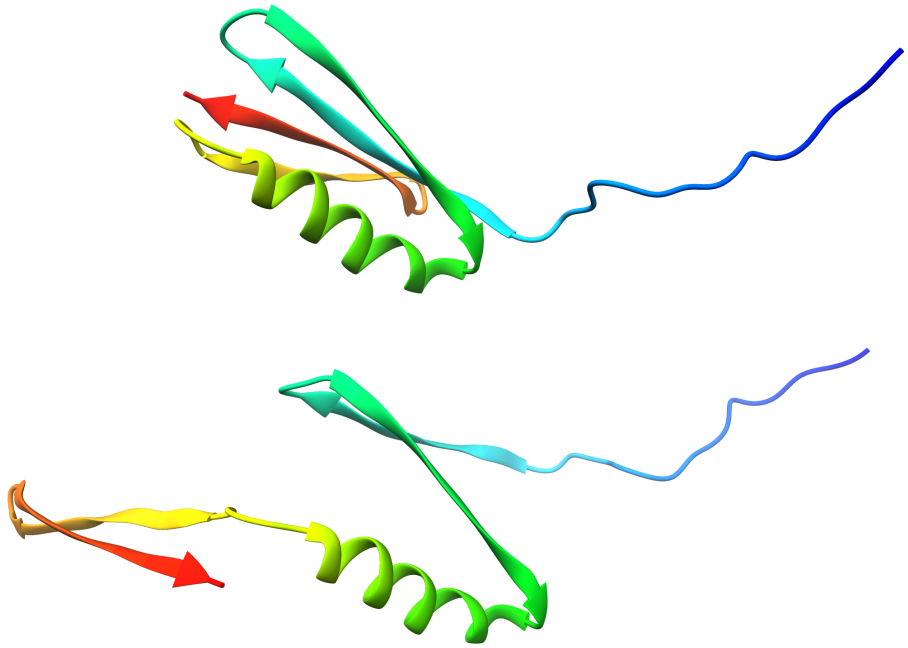

Figure A.5: Start and end conformations of the unfolding animation of 2PTL.