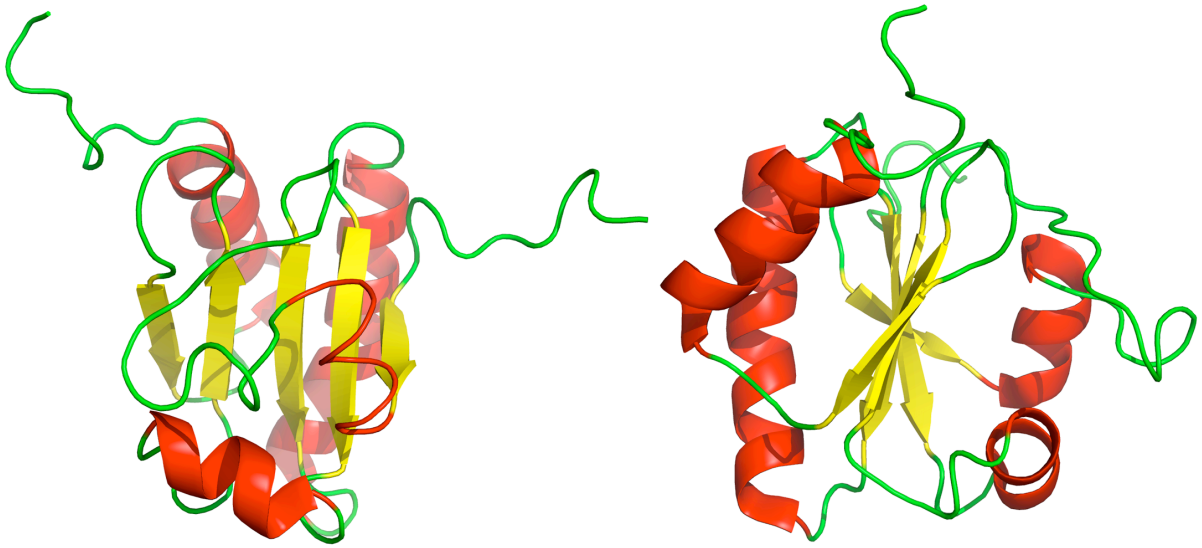
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Ph.D. thesis

Rasmus Fonseca

Reducing the Search Space in Protein Structure Prediction



Department of Computer Science
Academic advisor: Pawel Winter
Submitted: September 24th 2012

Abstract

The protein structure prediction problem is that of computationally predicting the three-dimensional structure of an amino acid chain from the sequence of amino acids alone. This has been an open problem for more than 30 years and developing a practical solution is widely considered the 'holy grail' of computational biology. While 'de novo' protein structure prediction is possible for some short chains, this thesis investigates how improved data structures and algorithms for representing and analyzing proteins can help predict the structure of large proteins faster and more reliably.

This thesis consists of eight research papers and a summary of their contents. Half the papers address the problem of efficiently exploring the space of protein conformations such that promising structures are more frequently sampled. For example we investigate how hard constraints, which are rarely used in molecular modelling, can be formulated and used to improve optimization methods used in structure prediction.

The other half of this work deals with efficient data structures for representing protein structures. When performing a conformational search atoms change positions and it is necessary to check for atom collisions very frequently. We present a data structure, based on bounding volume hierarchies, that decrease the computational time required to perform these operations by a factor of 3 compared to similar state-of-the-art methods. Another challenge when generating high-quality protein structures is to detect and remove packing flaws, i.e. small holes in the protein interior. We present a new method based on Delaunay tessellations that reduces the computational time of these detections four-fold compared to state-of-the-art methods.

Acknowledgements

Most of what I have achieved I owe to my supervisor, my wife and my dad. Pawel Winter has patiently supported me academically for almost five years, Maj Fonseca has patiently listened to my contemplations and heaps of ideas for almost a decade and Jan Riboe has patiently encouraged me in everything I do since before I can remember.

Additionally I am grateful to my collaborators - Glennie Helles Sindholt, Martin Paluszewski, Kevin Karplus and Desirée M. S. Jørgensen - for many interesting discussions.

Preface

I started my Ph.D. studies at Department of Computer Science (DIKU), University of Copenhagen in September 2009 under the supervision of Professor Pawel Winter. I finished my masters from DIKU a few months previously with experience in combinatorial optimization and applications of computer science to the field of bioinformatics. At this time our group had a strong profile in different network optimization and packing problems, but for a few years we had been looking at problems in computational biology. From September 2010 to June 2011 I visited Professor Kevin Karplus at University of California, Santa Cruz, as part of my Ph.D. studies. Kevin had for many years been the head of a large group doing research on protein structure prediction and sequence analysis and several previous Ph.D. students from DIKU had visited his lab.

This thesis consists of eight research papers and a summary of each paper. The summary is meant to be readable without necessarily going through the corresponding paper. In total there are 5 journal papers and 3 conference contributions, one of which is an extended abstract. All of the papers have been peer-reviewed. *The most important part of this Ph.D. thesis is the papers – the summary is written simply to assist readers that are unfamiliar with the field, to provide interesting background for our research and to extend the discussions in the papers.*

In the original Ph.D. proposal the goal of this thesis was summarized in the following way: "In this thesis I wish to examine different representations and develop improvements to existing or new representations that enable protein structure prediction methods to predict the native structure of large proteins more reliably". The problems that we have worked on all address this goal to some extent. The first chapter is a general introduction to the field of protein structure prediction, intended to make this thesis understandable to readers with limited knowledge of biology. The second chapter investigates how improvements to search strategies in protein structure prediction can be obtained by guiding the search and limiting the search space in different ways. The third chapter describes improvements to efficient protein representations and methods for analyzing protein structures using data structures from computational geometry. Conclusions and future directions are outlined in the fourth chapter and, finally, all papers are included in the fifth chapter.

Contents

Chapter 1

Introduction

Proteins are the building blocks of all living organisms. Approximately half of the dry weight of the human body is made up of proteins. Different proteins perform very different tasks, from maintaining the shape of cells to acting as neurotransmitters in the brain. A protein is synthesized using the genetic information encoded in DNA which contain a 'recipe' for generating a *sequence* of interconnected amino acids. This chain of amino acids folds to a three-dimensional structure called the *native structure*. The process of folding into the native structure is believed to be guided only (or at least mainly) by the sequence of amino acids (see Figure 1.1). Biochemical NMR and X-ray experiments can sometimes determine the native structure after the protein is folded, but each experiment is expensive and takes from days to months to complete. Still, many sequences are known and many new are generated every day for which we would like to determine the native structure. This could for instance give insights into diseases caused by misfolding proteins such as Alzheimer's disease, cystic fibrosis or Huntington's disease, but there are a plethora of different ways structural information can help biology, pharmacology and even engineering. The *protein structure prediction* problem is that of computationally predicting the native structure of a protein from the sequence alone and thereby provide this structural information. This has been an open problem for more than 30 years and developing a practical solution is widely considered the 'holy grail' of computational biology.

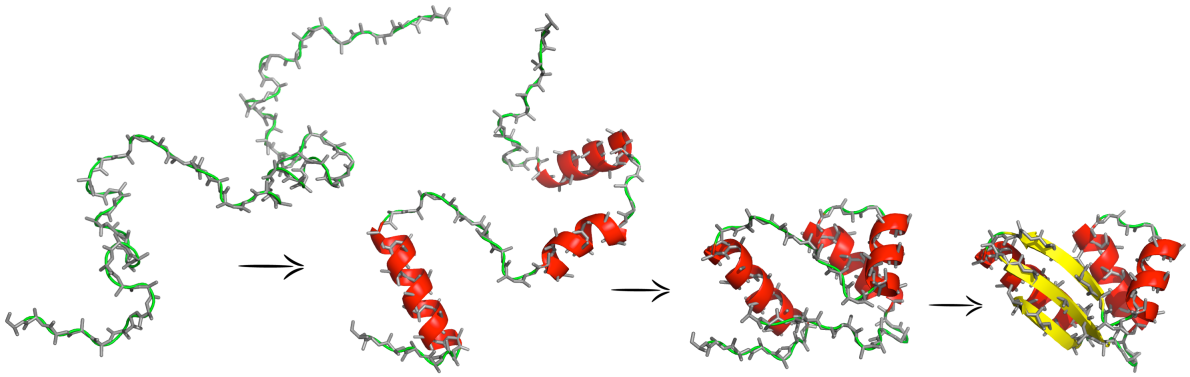


Figure 1.1: A protein folding from an extended structure to the native state. Only atoms that are part of the backbone are shown as well as the so-called ribbon diagram which indicates helical (red) and sheet-like (yellow) parts of the protein. Generated with PyMOL [1]. PDB-id: 1CTF.

1.1 Protein structure

A protein is a long chain of amino acids. Each amino acid consists of a nitrogen atom, a central carbon atom and a CO-group all bonded covalently together (see Figure 1.2). These atoms are named simply N, C_α, C and O. The chemical properties specific to an amino acid is determined by the side-chain (denoted R) which is covalently bonded to the C_α atom. In humans, the side-chain can be one of 20 different combinations of carbon, nitrogen, oxygen and sulfur[1] (see Figure 1.3). The atoms in the side-chain are named using their element and a Greek letter indicating the number of covalent bonds to the C_α atom. For instance, the first carbon atom is named C_β.

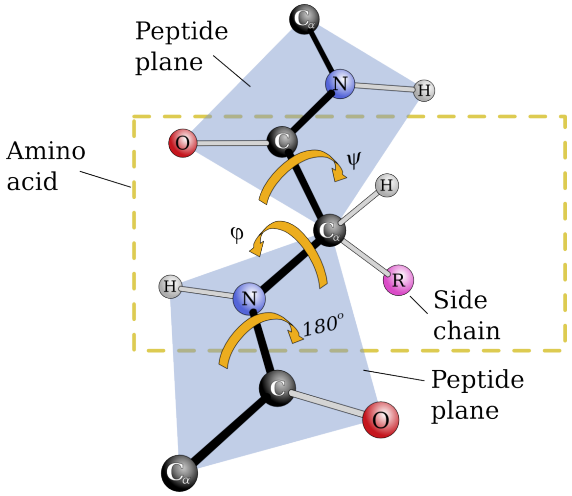


Figure 1.2: The structure of an amino acid. The dihedral angles (torsion angles) around the bonds adjacent to the C_α atom are called ϕ and ψ. Because of the partial double bond between the N and C atoms, the third bond typically has a fixed 180° dihedral angle.

When two amino acids bind to each other, a partial double-bond is created between the C-atom in the first amino acid and the N-atom in the second. When several amino acids are bonded like this they form a sequence where the amino acid with the non-bonded N-atom is defined to be the first (the N-terminal) and the one with the non-bonded C-atom is the last (C-terminal).

After being transcribed from the gene, the protein briefly exist as an extended sequence of amino acids surrounded by the solvent (mostly water). Within a few milliseconds the protein folds to minimize the Gibbs free energy. This process is mainly affected by two forces: Steric repulsion of electron clouds (atoms do not like to clash) and attraction of atom groups with opposite charge. The attraction between charged atoms creates the energetically favorable *hydrogen bonds* which are positively charged NH-groups interacting with negatively charged oxygen atoms. It is very common for the backbone CO-group to interact with the backbone NH-group which is located four amino acids further along the chain. When this happens to all amino acids in a long stretch, a helical shape, called an α-helix, occurs (left side of Figure 1.4). It is also possible for the NH-CO interaction to occur between stretches of amino acids far removed in the sequence, forming slightly twisted sheet-like structures called β-sheets (right side of Figure 1.4). α-helices and β-sheets occur in almost all proteins and are referred to as *secondary structures*.

[1] For simplicity hydrogens are currently disregarded.

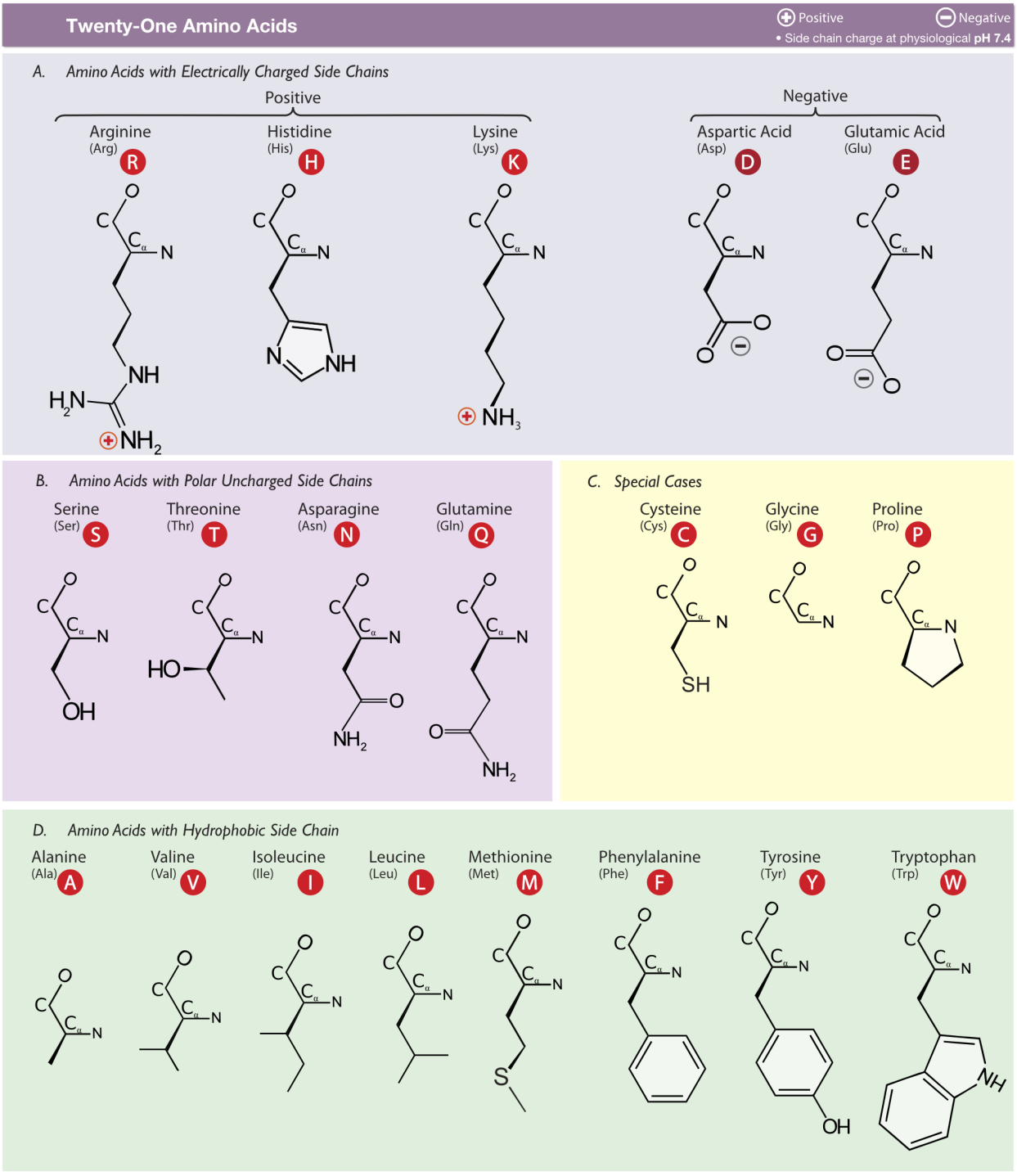


Figure 1.3: The 20 standard amino acid side chains with the backbone on the top. Unlabeled vertices are carbons. Modified version of `http://en.wikipedia.org/wiki/File:Amino_Acids.svg`

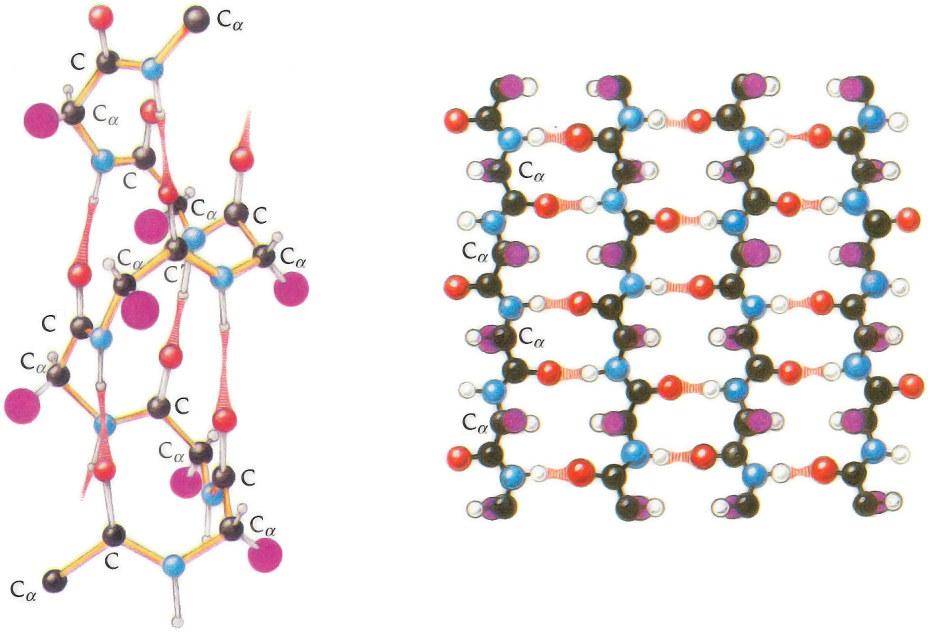


Figure 1.4: Left: An α-helix with backbone marked yellow and hydrogen bonds marked red. The backbone CO-group in amino acid i forms a hydrogen bond with the backbone NH-group in amino acid $i + 4$. **Right**: A β-sheet. The backbone CO-group form hydrogen bonds with the backbone NH-group in a different part of the chain. Slightly modified illustration from [2, p. 15-18].

Another consequence of the attractions between charged atom-groups is the so-called hydrophobic effect or *hydrophobic burial*. Some side-chains contain only neutrally charged carbons, so if one of these side-chains is in contact with the solvent it will prevent dipolar water molecules from interacting with each other. It is therefore energetically favorable for such side-chains to avoid water (hence they are called hydrophobic) by packing against each other in the center of the protein, shielded from the solvent. Charged side-chains containing O and NH-groups, on the other hand, have a high propensity for forming hydrogen bonds with each other or with the surrounding water molecules (hence they are called hydrophilic).

1.2 Protein structure prediction

The *protein folding* and protein structure prediction (PSP) problems both seek to predict the native structure computationally given only the sequence of amino acids. The prevailing belief is that the space of all feasible structures of the protein and the Gibbs free energy forms a funnel-shaped *energy landscape* that guides the protein structure to the native structure (see Figure 1.5). The native structure will correspond to the global minimum in the energy landscape or possibly to the minimum with the widest basin of attraction. Protein folding simulations seek to determine path-ways from an unfolded structure to the native structure, whereas PSP simply searches the energy landscape in any way possible to find the native structure.

A major problem for both these methods is to design an artificial *energy function* and a representation of the protein structure that results in an artificial energy landscape with similar properties

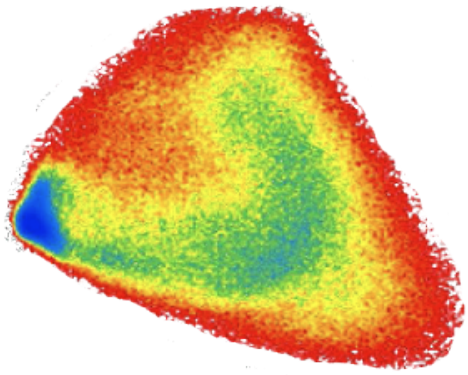


Figure 1.5: Visualization of the free energy landscape associated with the SH3 domain (PDB-id: 1SHG) computed by SciMAP. Blue is the lowest energy and red is the highest. The red parts correspond to unfolded structures while the leftmost blue region is the native structure. Image is originally from [3], but permission to reuse was granted in [4].

as the real energy landscape. Good physics-based energy functions combined with all-atom representations are computationally requiring so approximations are often needed. These are introduced via reduced representations of the protein where some atoms are disregarded or groups of atoms are considered one large atom. These approximations, however, change the energy landscape so multiple minima and barriers occur. Many PSP methods overcome this obstacle by trying to generate large amounts of low energy structures (*decoys*) in all the widest minima. One single decoy from each minimum is then *refined* using more accurate and requiring energy functions and protein structure representations. Hopefully, the native structure can finally be identified as the refined structure with lowest energy.

The current state-of-the-art PSP methods can predict the native structure of proteins containing up to 100 amino acids [5] without assuming that a similar protein is known (referred to as *de novo* PSP). These predictions require from days to years of CPU-time. A typical protein, however, contains around 300 amino acids [6] and some proteins contain several thousands. Improvements to protein representations and prediction methods are therefore important to make predictions of all protein sizes feasible.

1.3 Why 'de novo' protein structure prediction

Methods for solving the PSP problem are generally divided into two types, comparative methods and 'de novo' methods. When predicting the structure of a particular sequence, comparative methods will assume that a similar protein structure has previously been determined and is located in a protein database [7]. The challenge for these methods is to locate this structure and refine it. De novo methods attempt to predict the native structure using either physical or statistical knowledge of how proteins fold and thereby perform a conformational search starting, in principle, from a completely stretched out chain structure.

The most successful methods, so far, are the comparative methods, as witnessed every second year at the CASP experiments [8]. It is therefore necessary to briefly outline the reason why our research focuses on de novo methods when they seem inferior to the comparative methods that

solve the same problem.

The reason for focusing on de novo methods is three-fold. First, some protein structures have not previously been solved experimentally, so a comparative method would not be able to locate them in any database. Second, a de novo method that solves the PSP problem will explain much more about how proteins actually work than any comparative method. It will provide information on which effects should be included in the energy function, which effects are more important and possibly tell something about the dynamic properties of macromolecular systems in general. Finally, a successful de novo method would be adaptable to many other molecular modelling problems such as RNA-folding, protein design, macromolecular docking or function prediction.

1.4 Definitions

The secondary structure elements of proteins are normally called α-helices, β-strands and β-sheets. Some sections refer to these terms quite often, so α and β will be omitted and they will be referred to simply as helices, strands and sheets.

The terms protein conformation and protein structure are somewhat synonymous. Protein conformation will typically refer to a dynamically changing representation of protein structure, while a protein structure is a static set of atom positions.

Chapter 2

Improving the search strategy in protein structure prediction

2.1 Backbone torsion angles in coil segments

In many metaheuristic optimization methods the protein conformation is represented by a torsion angle for every rotatable covalent bond (see Figure 1.2). Given a sequence of amino acids, $i = 1 \ldots n$, each amino acid has 2 backbone torsion angles, ϕ_i and ψ_i, and from 0 to 4 side-chain torsion angles, $\chi_i^1, \ldots, \chi_i^4$. Together, these torsion angles specify the conformation of the protein. Metaheuristics can be improved significantly by including prior knowledge of probable torsion angles. The backbone torsion angles (ϕ_i and ψ_i) are particularly important since they determine the overall shape of the proteins.

The first step in predicting (ϕ_i, ψ_i)-angles is to determine the secondary structure. It is possible to predict the secondary structure using the amino acid sequence as input to any of the popular machine learning methods such as neural networks or hidden Markov models. The best approaches predict roughly 80% of the amino acids correctly, which is considered reasonably accurate (see e.g. [10] for a recent review). It is well-known that the hydrogen bonds between amino acids in helices and sheets cause them to adopt very particular combinations of (ϕ_i, ψ_i)-angles (see Figure 2.1). It is therefore not difficult to predict torsion angles for amino acids in strand or helix segments. But the remaining amino acids, those located in so-called *coil*-regions, are typically not constrained by hydrogen bonds and torsion angles and will not follow an easily recognized pattern. The goal of our paper "Predicting Dihedral Angle Probability Distributions for Protein Coil Residues" (included in Section 5.1) was therefore to investigate how well the torsion angles for amino acids in coil regions could be predicted.

Our approach was to use windows of 7 adjacent amino acids as input to a trained feed-forward neural network in order to predict the (ϕ, ψ)-angles of the amino acid in the center of each window. The window could span both helix- and strand-segments, but the central amino acid was always a coil. The output of the neural network was discretized by mapping pairs of (ϕ, ψ) angles to 144 non-overlapping $30° \times 30°$ bins in the Ramachandran plot (see Figure 2.2). The neural network was trained using ordinary backpropagation on a large training set extracted from the PDBSelect25 data set [11].

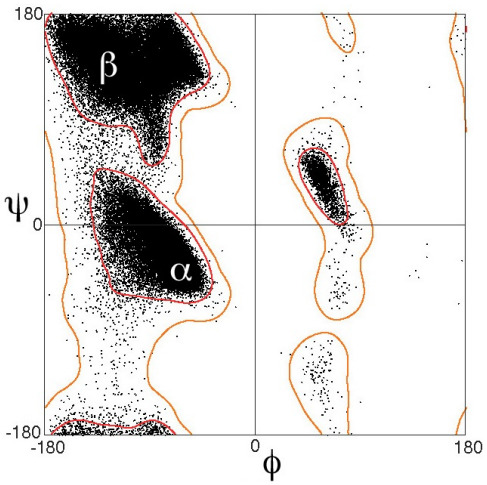


Figure 2.1: A typical Ramachandran plot showing (ϕ, ψ)-combinations for 100.000 amino acids. Amino acids in helices will occupy the α-region, while those in sheets will occur in the β-region. The completely empty regions correspond to (ϕ, ψ)-angles that cause clashes between atoms on the backbone. Image from [9].

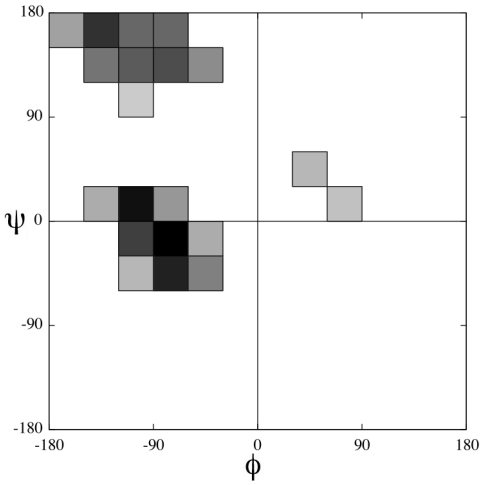


Figure 2.2: The 20 bins with highest output value from our trained neural network given the amino acid sequence "ELDTEDA". Each bin corresponds to a $30° \times 30°$ area of (ϕ, ψ)-angles for the central threonine amino acid (the "T" in the sequence).

Our first experiment was to evaluate how often the correct (ϕ, ψ)-pair fell within the bin with highest output-value. A predictor that only considers a single amino acid, i.e. a neural network with a window size of 1, was used as a null-model. Compared to this null-model our prediction method was an improvement. Particularly the prediction of backbone torsions for hydrophilic amino acids was improved significantly. This effect is probably observed because hydrophilic amino acids, which are more often exposed to the solvent, have more flexibility and are therefore more easily affected by adjacent amino acids. Hydrophobic amino acids, buried in the core, are to a higher degree, affected by tight packing of atoms that are not necessarily adjacent in the chain.

Despite the improvement over the null-model, the accuracy of single-bin predictions was still only about 15%. It was therefore investigated how often the correct (ϕ, ψ)-pair fell inside one of the x bins with highest output-value from the neural network, where x was varied from 1 to the total number of bins. It was observed that setting $x \simeq 20$ ensured that the correct bin was among the x around 80% of the time. This accuracy is comparable to that of secondary structure predictions and hence estimated to be sufficient for sampling realistic backbone torsion angles in coil-regions.

After submitting the paper, the neural network was incorporated into the search strategy of an all-atom PSP method based on genetic algorithms in order to improve the sampling of torsion angles in coil-regions [12]. First, output-values for the bins were normalized so they all summed to 1. A bin was then chosen probabilistically based on the normalized value, and a random angle-pair within this bin was picked. The effect of incorporating our neural network was, for example, a doubling in the number of generated decoys with a high similarity to the native structure (RMSD [13] less than 4Å) for the crambin protein (see Figure 2.3).

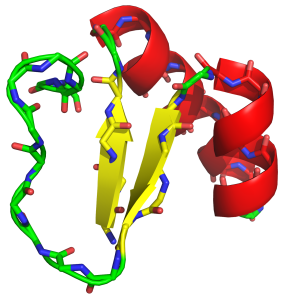


Figure 2.3: A ribbon illustration of the 46 amino acid crambin protein (PDB id: 1CRN).

One of the challenges with this project was to find an adequate way of comparing our method with similar reported results. Boomsma et al. [14], for instance, published a paper just before we submitted ours which presents a framework that uses hidden Markov models to sample torsion angles for any type of amino acid (not just coils). They generated 100 (ϕ, ψ)-samples for each amino acid and reported the smallest distance from any of these samples to the (ϕ, ψ)-pair of the native structure within the Ramachandran plot. However, their results are not comparable to ours since they are generated for all types of secondary structure and not just coil, which is the hardest and most interesting to predict.

Kuang et al. [15] and Zimmermann and Hansmann [16] used methods similar to ours but instead of having small bins, from which meaningful samples can be drawn, they included very few large bins that cover all of the allowed parts of the Ramachandran map. In the paper we attempted to make a comparison with these results, but the comparison is slightly unfair since the binning of the Ramachandran plot is very different. One interesting direction for future work would therefore be to acquire all relevant pieces of software that can be used to sample torsion angles and do a proper comparison between their performance. Since torsion angle pairs in secondary structures follow much simpler patterns, the comparison should of course distinguish between amino acids in helices, strands and coils.

One important problem with comparing the sampling methods is to evaluate their applicability in protein conformational sampling. This could be achieved by seeking an optimal combination between a representative structural model and different torsion sampling methods. Several other elements of torsional sampling can then be investigated, such as the optimal overall framework (Gaussian mixture models or drawing sampling from bins) or the best placement and shape of bins in the Ramachandran map.

2.2 Finding protein decoys using branch and bound

The Efficient Branch and Bound Algorithm (*EBBA*) is a method developed by Paluszewski and Winter for generating protein decoys [17, 18]. In EBBA the protein structure is modelled in such a way that the entire search space can be exhaustively explored for proteins of a reasonable size. This approach is in many ways unique, and it is important to some of the ideas in this Ph.D. thesis. The following sections briefly introduce EBBA.

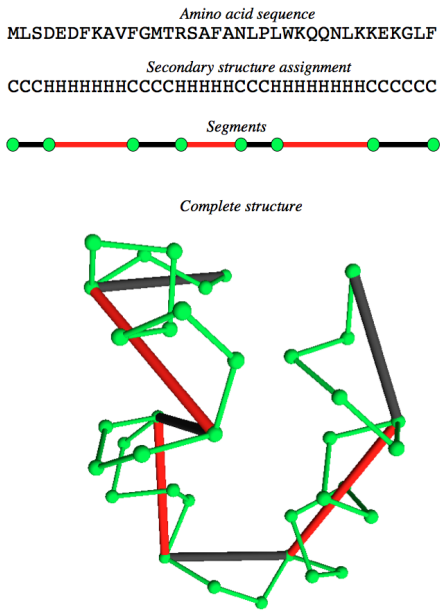


Figure 2.4: Example of C_α-trace in EBBAs representation. Each segment has a discrete number of directions and rotations. The sequence and secondary structure are from the Villin headpiece (PDB id 1VII). Image from [18].

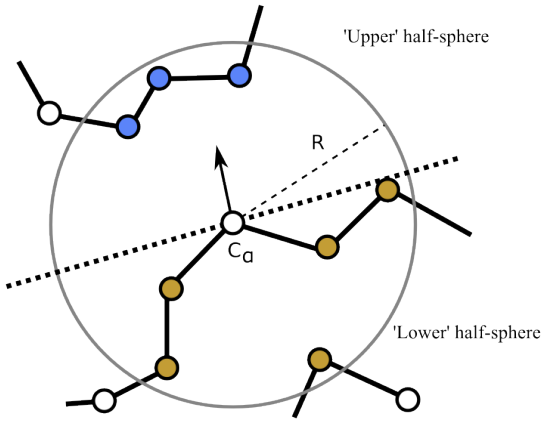


Figure 2.5: A simplified C_α trace, and an example illustrating the half-sphere exposure of the central amino acid. The up/down pair for this amino acid is (3, 5), and the contact number is 8. All amino acids are assigned up/down numbers in this manner.

The input to EBBA is an amino acid sequence and a secondary structure prediction. The local structure of each segment is fixed but the segment can assume a discrete number of directions and rotations (see Figure 2.4). Only the central C_α atom of each amino acid needs to be defined in this representation. The first C_α in the first segment is placed at origo, while the first C_α in the remaining segments are placed appropriately near the last C_α in the previous segment. Given m segments that each can assume r rotations and d directions, the search-space has a size of $(d \cdot r)^m$. This can be reduced to $(d \cdot r)^{m-1}$ by fixing the direction and rotation of the first segment. For the proteins that were tested, the number of segments was between 5 and 11, d was 12 and r was 8. A typical protein, therefore, had a search space with size at least 10^{13} – much too large to be searched exhaustively.

When the direction and rotation of each segment were fixed, the C_α atoms had well-defined positions. An energy-measure, based on half-sphere exposure (HSE) [19], was then used to estimate the quality of the entire protein structure. In essence, HSE is a list of numbers, two for every amino acid, indicating how many other amino acids are in the half-sphere 'above' the C_α atom and how many are 'below' (see Figure 2.5). The HSE of a protein can be predicted from the primary sequence [20], and the energy-measure in EBBA indicated how much the HSE from a certain structure deviate from the predicted HSE.

Given a partial structure, i.e. a structure where only some of the first segments have been placed, EBBA had a routine to efficiently determine how good the HSE-based energy-measure could possibly hope to get if the search proceeded from that partial structure. This *lower bound*

makes an explicit search of all possible solutions unnecessary. If the search had already encountered a complete protein structure with energy, E, and was considering a partial structure whose lower bound was larger than E, then no completion of the partial structure could be optimal and it could therefore be disregarded completely. This is the essence of branch and bound algorithms. Following this approach EBBA generated a set of decoy structures whose HSE deviated only very little from the predicted. A high-quality structure (RMSD less than 6Å) could always be found among these decoys.

EBBA is one of very few methods that attempt to attack PSP using exact methods[1]. Almost every other method uses some form of metaheuristic where no guarantee about the optimality of the results can be given (with limited computational time). Another interesting property of the branch and bound method is that it benefits greatly from hard constraints. In metaheristic methods, constraints are typically relaxed and incorporated into the objective function.

2.3 Bee colony optimization for generating protein decoys

EBBA often needed to run for several hours before the first complete structure was found and pruning of partial structures could begin. It was therefore examined if a metaheuristic could be used to quickly find a suitable complete structure and thereby help EBBA prune earlier in the search. The Bee Colony Optimization (BCO) metaheuristic was invented a few years previously and it seemed like a promising approach for this problem. Our paper "Protein Structure Prediction Using Bee Colony Optimization Metaheuristic" (included in Section 5.2) describes how this metaheuristic was adapted to generate decoys using EBBA's discretized representation.

The BCO metaheuristic explores the search space in a manner similar to how honey bee colonies forage. In a bee hive, scout bees randomly search for promising flower patches. At regular intervals they return to the hive and perform a so-called 'waggle dance' which indicates the location and potential of the flower patch they found. Worker-bees in the hive observe these waggle dances and based on the indicated potential of each flower patch they make a probabilistic decision to go explore one of them and bring back nectar. In the context of metaheuristics, the waggle dance can be interpreted as a way of determining the intensity of local search near a particular solution (i.e. exploitation). Similarly, the number of scout bees can be interpreted as a parameter indicating how many computational resources should be spent on randomly searching for new promising solutions (i.e. exploration).

Using the EBBA model a variant of BCO was implemented with the purpose of generating decoys. One major problem with this approach was that the EBBA model had many constraints limiting the search-space. This was good for branch and bound, but it was exceedingly hard for BCO to find even a single feasible solution. We tried increasing the number of possible directions and rotations and to split long coil-segments in the middle. This had the effect that the search-space increased dramatically and finding a feasible solution became possible. However, a solution in this *relaxed model* was not necessarily an upper bound on the optimal solution in the original EBBA model making it unusable for pruning.

[1]Some exceptions include the exact methods solving the simplified HP-lattice problem [21]. This problem, however, has only a limited connection to PSP and solutions are not easily interpreted as realistic protein structures. Another notable example is the αBB method [22] based on the astro-fold system.

The metaheuristic was tested on the relaxed model by comparing it to a random-restart simulated annealing (SA) method. SA is a commonly used metaheuristic inspired by the metallurgical process of annealing a material to optimize certain properties. A temperature parameter controls whether SA behaves like a random walk (high temperatures) or like hill-climbing (low temperature). The temperature is then slowly lowered which guarantees a good trade-off between exploration at high temperatures and exploitation at lower temperatures. By comparing the lowest observed energies during conformational search we concluded that BCO significantly outperformed the SA metaheuristic using the relaxed model. We also compared the lowest observed energy in the relaxed model with the optimal value in the original model found using EBBA. Here we also observed a significant improvement.

The SA method was allowed to restart 10 times during the optimization while BCO was allowed to send out scouts for every iteration. This probably gave BCO an advantage, and given more time it would have been preferable to test the BCO method more fairly against other types of metaheuristics. For instance, it might have been interesting to see how well it would have compared to a very basic random restart hill-climbing that was restarted every time a solution converged. Such a method would restart more frequently and possibly perform better than the SA method.

For most of the problem instances BCO generated decoys that were closer to the native structure than those found using either SA or EBBA. Two proteins from the CASP7 experiment were evaluated and BCO generated structures that were comparable in quality to those from the best prediction groups.

In November 2005 two metaheuristics, both based on the foraging behaviour of bees, were described in technical reports [23, 24] by two different research groups. There are some technical differences but essentially they describe the same approach. Several conference and journal publications describing both the algorithms and different applications of each variant have since been published but neither research group ever cites the other. One of the achievements of this work was a hybrid description of the BCO method that encapsulated both the methods.

The need for relaxing EBBA's representation indicate that the metaheuristics do not work well with constraints. Since metaheuristics depend on being able to step from one local minimum in the energy landscape to another, constraints that are not carefully designed to aid the permutation operators will hinder its local search significantly. Too extensive search spaces, however, are also problematic for optimization methods. Hence, it can be concluded that for constraints to be of any use to metaheuristic search methods they must be an implicit part of the representation.

2.4 Using β-sheets in conformational search

Reducing the search space by incorporating constraints into the representation of the PSP problem has been done before. Molecular dynamics methods represent molecules using the position and velocity of individual atoms. Covalent bonds between atoms are treated like springs [25], and even small displacements of a single atom can increase the energy term describing this spring significantly. An alternative is to incorporate the covalent bonds into the representation such that two bonded atoms are always the same distance from each other (corresponding to an equality constraint). Changes to the structure are then introduced by rotating parts of the protein chain around covalent bonds.

Fragment assembly can also be seen as an inclusion of parts of the energy function in the representation. For each amino acid there are two rotatable backbone bonds (ϕ, ψ), but many combinations of these angles are unfavoured or impossible because of the geometry of the backbone. This can be modelled either by adding a term to the energy function or, as in fragment assembly, by using pre-calculated fragments of backbone (typically 5-7 amino acids) with realistic sets of backbone torsion angles. The constraints preventing (ϕ, ψ)-angles from entering illegal parts of the Ramachandran plot (see Figure 2.1) are therefore implicitly represented by the choice of valid fragments. Secondary structure predictions can further improve fragment assembly by affecting which types of fragments to test.

The above improvements all decrease the number of degrees of freedom. The energy landscape becomes smaller and less rugged making it possible to introduce large structural changes without always increasing the energy dramatically.

Sheets are another important secondary structure formed during the folding of proteins. Two stretches of backbone from different parts of the chain align and form hydrogen bonds between their backbone CO- and NH-groups. If a protein contains sheets, and assuming that it is possible to predict which amino acids form hydrogen bonds in the sheet, the conformational search space of a protein can be significantly reduced. Unlike the reductions to the search space mentioned above, predicted sheets are not frequently used to reduce the number of degrees of freedom.

One of the hypotheses of this dissertation is that the conformational search space of proteins with sheets can be significantly reduced by incorporating predicted sheets in the representation of protein structures. The following subsection outlines our suggestion to predict sheets and the last subsection describes how such predictions can be used in conformational search.

2.4.1 Enumerating β-topologies

The first step in determining how strands form pairs is to predict the location of the strands themselves. As mentioned previously there is a plethora of different machine learning methods that attempt to predict the secondary structure (see e.g. [10]). Each of these output the locations of both helices and strands.

With the strand locations fixed, the next problem is to predict which strands form pairs. A complete specification of which strands form pairs, and whether each pair is parallel or anti-parallel is here called a *β-topology* (see Figure 2.6 for examples). There are some methods that attempt to predict β-topologies [26, 27, 28, 29, 30]. While they often manage to predict a couple of strand pairs, correctly determining the entire β-topology is more challenging. Since we wish to use the entire β-topology as a constraint it is very important that all pairs are correctly predicted, otherwise the native structure might be excluded from the conformational search. We therefore propose a slightly different approach based on the following three steps:

1. Given the secondary structure, enumerate all possible β-topologies
2. Assign a score to each β-topology indicating its probability of being correct
3. Extract enough of the highest-scoring β-topologies such that the correct one is guaranteed to be among them.

One or several decoy structure can then be generated for each of the extracted β-topologies. Since one is guaranteed to be correct, at least one of the decoys will be generated with a very accurate set of constraints. Furthermore, generating each decoy will be extremely fast if a framework can be

created that adequately takes advantage of such constraints. The conference paper "Ranking Beta Sheet Topologies of Proteins" (included in Section 5.3) outlines this idea and seeks to evaluate how best to carry out the three steps mentioned above. A second paper titled "Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction" (see Section 5.4) further attempts to improve the method by taking the problem of inaccuracies in the secondary structure prediction into account.

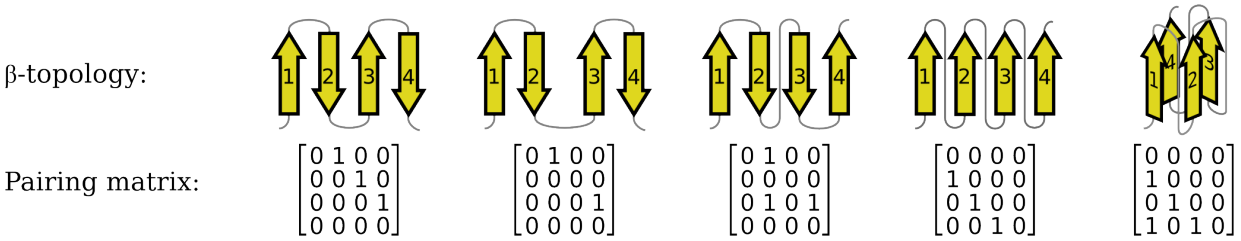


Figure 2.6: Five valid β-topologies (out of 156 possible) for a protein with four strands. A 1-entry at the ith row and jth column of the pairing matrix indicates that strands i and j are paired. If i is larger than j they are parallel and anti-parallel otherwise. Multiple sheets and barrels are possible in this representation.

A β-topology is represented by a binary $m \times m$ *pairing matrix*, where m is the number of strands in the secondary structure. If an entry $a_{ij} = 1$ then strands i and j are paired. If $i > j$ (i.e. the 1 is below the matrix diagonal) it is a parallel pair, and if $i < j$ (above the matrix diagonal) it is an antiparallel pair. A *valid pairing matrix* is a pairing matrix where each strand has at least one partner, at most two partners and is not paired with itself. All possible β-topologies could then be generated by enumerating all valid pairing matrices.

To assign a score to a β-topology we implemented and compared two different methods from the literature. The first method used a combination of neural networks and dynamic programming to determine how likely it is for two strands to be paired [27]. The score of the entire β-topology is the average of these probabilities. The second method combines features such as loop lengths, the number of strand amino acids and the number of sheets in a purely statistical method that assigns a probability to a β-topology [31].

To determine a sufficient number of top-ranking β-topologies to extract, a large set of proteins was investigated. For each protein the rank of the β-topology corresponding to the native was recorded using both scoring methods.

The main achievement of this paper was the outline of an approach to enumerate β-topologies and to use them in the conformational search. The remaining results attempt to illustrate how feasible such an approach can be. For instance, we investigated the average and median ranks of the correct β-topology for proteins containing different numbers of strands. The conclusion was that for most proteins with 2-4 strands it was enough to only consider the 3 highest ranked β-topologies. For proteins with more strands it became necessary to look at a few hundred β-topologies before it could be guaranteed that the correct one was in the set. However, since typical PSP methods generate thousands of decoys it is still feasible to generate several decoys for each β-topology, even assuming that the search is not sped up by the inclusion of constraints.

As mentioned we compared two methods for scoring β-topologies. For a majority of proteins,

particularly those with 5 or more strands, the scoring method by Cheng and Baldi [27] outperforms that of Ruczinski et al. [31]. This conclusion was based on the median rank of the correct β-topology.

Guaranteeing that the correct β-topology was among the generated had two problems. First, there was a combinatorial explosion in the number of possible β-topologies for proteins containing more than 7 strands. Fortunately, most single domain proteins have fewer strands, but for the remaining proteins this was a considerable problem. Our solution was to look for 'almost correct' β-topologies, i.e. β-topologies containing a subset of the strand pairs that are in the correct β-topology (in our matrix-representation an almost correct β-topology is a minor of the correct one). These almost correct β-topologies provide a set of constraints for PSP that are almost as good as those obtained from the correct one, and they can still be enumerated when proteins have too many strands.

The second problem with guaranteeing the existence of the correct β-topology was that secondary structure predictors tend to underpredict strands. This issue was addressed by finding alternative locations for strands and enumerate all possible secondary structures. For each secondary structure all β-topologies were then generated. This approach helped guarantee that the correct β-topology was among the generated, but it also significantly increased the number of β-topologies.

There are a number of papers that attempt to predict which strands form pairs. Predictions of attributes such as domain boundaries and burial depth might be used to improve such methods further. Additionally, it has been established that pairings of strands that are far removed in the chain only occur after a series of more local strand pairings have brought the two strands close to each other [32]. This concept of a folding pathway has been used in β-topology prediction [28], but methods that use all the available auxiliary information might improve the scoring of β-topologies further.

We are currently working on a new representation of protein structures that takes advantage of β-topologies. This work is in progress and the following section outline our approach as an extended conclusion to the papers discussed in this section.

2.4.2 Using predicted β-sheets in conformational search

A few existing PSP methods have previously used β-topologies in conformational search. The following section describes these methods briefly and finally our own, still unfinished, approach is outlined.

The BuildBeta system [33] takes as input the primary structure, secondary structure and β-topology. It starts with an elongated chain with fixed torsion angles in strands and helices. Each strand-pair in the β-topology is then *zipped*, i.e. brought together so the strands align. The zipping is performed using an inverse kinematics method based on transposed Jacobians to change the torsion angles in the intermediate coil-regions. After zipping all strand-pairs, helices that clash with the sheet are moved, also using inverse kinematics. SCWRL4 is used to optimize side-chain rotamers, but other than that no refinement is done to remove clashes or undesired knots. Since the process is not deterministic it can be repeated multiple times to obtain a set of different decoy structures. The system was tested on 10 proteins with 85-187 amino acids containing 4-6 strands. All possible single sheet β-topologies were used in turn and a set of decoys were generated for each. Most decoys took between 6 and 20 seconds to generate, and the lowest RMSD among the generated decoys was around 5.6Å. This RMSD is surprisingly good considering the size of the

proteins and the fact that no conformational search was performed. Figure 2.7 illustrates how close to the native structure some of the best decoys were. The sheets in the chosen proteins have a very

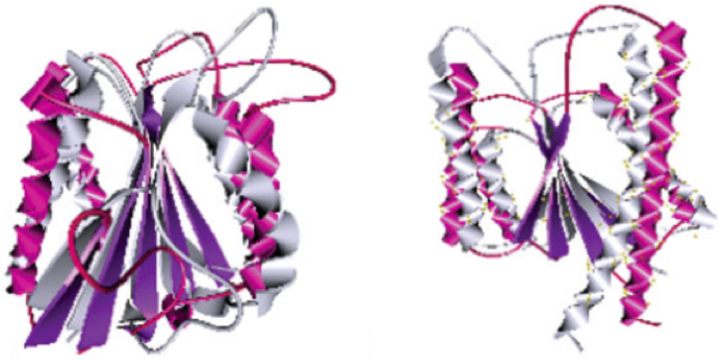


Figure 2.7: Two examples of structures generated with BuildBeta (colored) superposed onto their native structures (gray, PDB-ids: 1F4P and 1DI0). Figures are from [33].

regular twist and were therefore ideal for this type of experiment. The reason for the good results might therefore be that the proteins were hand-picked or perhaps they simply reflected the best results from a larger test-set.

Porwal et al. [34] describes a molecular dynamics approach to protein folding based on the TINKER-framework using the CHARMM19 energy function [35]. Here the β-topology is enforced by adding a so-called *restraint* to the energy function which is a term that increases significantly when the atoms of two paired strands are too far apart[2]. The protein structure goes through several iterations where the torsion angles of a random coil amino acid are changed and then the structure is reoptimized. The method was compared to a variant of the same method, only without the restraints from the correct β-topology. Fifteen proteins with sizes 31-150 amino acids were tested and the similarity to the native structure (measured using SSAP [36, 37]) and energy was recorded. By including the β-topology restraint there was a small but consistent improvement in the similarity to the native structure, and a significant improvement in the lowest observed energy (typically around -50kcal/mole).

Bradley and Baker [38] also attempt to enforce pairing of strands in the conformational search in Rosetta. Strands are placed together and the loop-regions connecting strands are cut open. This results in a representation of the protein structure as a tree where internal nodes correspond to non-local contacts and leaves correspond to loop-ends. The closure of the loops is then built into the energy function. The new method was tested on 12 proteins with sizes 68-125. A large number of decoys was generated for each protein and the five most promising decoys were selected by picking the centroids of the five largest clusters. Except for three of the largest proteins the best of these five centroids had a significantly higher similarity to the native structure than the original fragment assembly method (evaluated using GDT [39, 40] with a distance cut-off of 4Å).

None of the methods mentioned above used β-topologies as real constraints in the conformational search, though some used them as relaxed constraints (restraints). The input to our own method is an amino acid sequence, predicted contact numbers and a β-topology. A discretized

[2]Though not written explicitly it is most likely a spring term.

representation similar to that of EBBA is used, but the structure of the sheet is specified in the representation.

For simplicity, assume initially that there is only one sheet. The sheet-structure is obtained by aligning each strand-pair and then placing backbone atoms ideally along a minimal surface (see Figure 2.8). Sheets in β-barrel and β-sandwich proteins have been shown often to behave as a minimal surface [41, 42]. A limited number of sheet-structures can be obtained by choosing different promising strand-pair alignments or different curvatures of the surface.

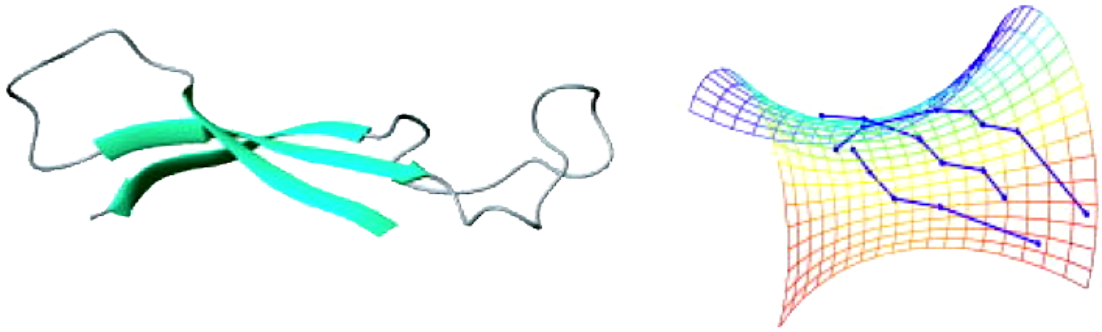


Figure 2.8: A minimal surface (catenoid) fitted to the sheet in 1AXI. Image is from [41].

Next, the coil/helix-segments connecting the strands are placed one at a time starting from each segments N-terminal. Each segment has a limited number of possible rotations and directions. Every time a segment has been placed a lower bound on the contact number energy is calculated. If this lower bound is higher than the energy of a previously found structure, or if the C-terminal of the segment is so far from the end of the loop-region that it can never be closed, the search is stopped for this partial structure.

The lower bound on the contact number is calculated by maintaining a map of possible (P), impossible (I) and certain (C) contacts in the structure. At first, all pairs of amino acids are in P. Amino acid pairs that are adjacent in the chain are moved to C, and after placing the sheet all pairs within strands can be moved to either C or I. After placing each segment further pairs can be moved from P to C or I and a lower bound on the contact number can be computed as

$$\mathcal{LB} = \sum_a \mathcal{LB}(a)$$

where the sum is over all amino acids, and

$$\mathcal{LB}(a) = \begin{cases} \max\left(CN_{\text{pred}}(a) - C(a) - P(a), 0\right) & \text{if } C(a) < CN_{\text{pred}}(a) \\ C(a) - CN_{\text{pred}}(a) & \text{otherwise} \end{cases}$$

Here $CN_{\text{pred}}(a)$ indicates the predicted contact number of a, $C(a)$ the number of contacts with a that are certain and $P(a)$ the number of potential contacts with a.

The final pieces for the lower bound method are currently being implemented and we are trying to extend it to use predicted half-sphere exposure as a lower bound as well. There is a number of interesting tree-theoretic and geometric challenges involved in this. For instance, the calculation of the lower-bound for each node in the branch and bound tree needs to query the contact-maps

for information about possible, impossible and certain contacts. When placing a segment, only the status of possible contacts can change, which means that a lot of computations can be spared by using the information of a parent node in the branch and bound tree. Since the contact-map has size $\Theta(n^2)$ it is impossible to store it in every node (or even every leaf) of the tree. Therefore, each node needs an efficient method to store only the status of contacts that changed. It should be analyzed how fast such queries can be performed by traversing the branch and bound tree.

Another challenge is to determine if there is a certain or impossible contact between an amino acid that has not yet been placed, a_u, and one that has, a_p. For a_u one can determine the *hinges*, i.e. the nearest placed amino acids along the chain in both directions. The set of possible placements for a_u will be in the intersection of two spheres centered at these hinges. The radius of each sphere is the largest possible distance, along the chain, from a_u to the hinge (see Figure 2.9). We denote the lens-shaped intersection of the spheres, L, and place a sphere, S, with a_p as center and the contact distance as radius. If L is contained in S then there is a certain contact between a_p and a_u. If the intersection of L and S is empty then it is impossible for a_p and a_u to be in contact. There are similar geometric predicates needed when determining the contact-status of two unplaced amino acids.

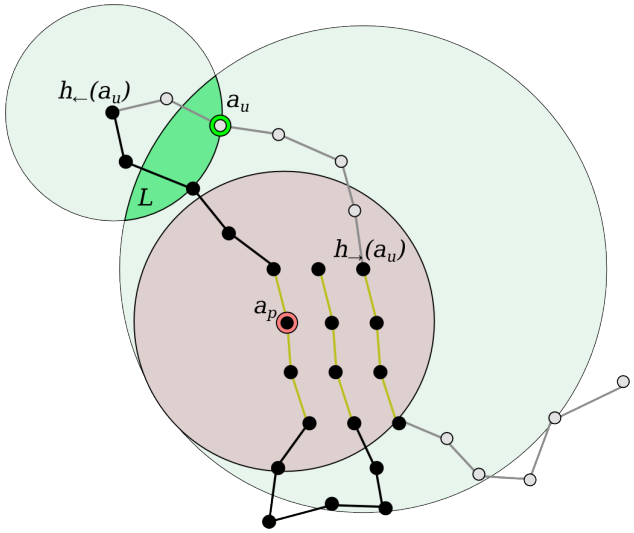


Figure 2.9: The geometric objects involved in determining if the placed amino acid a_p has a certain or impossible contact with the unplaced amino acid a_u. The hinges (nearest placed amino acid along the chain) of a_u are both the center a sphere having the largest possible distance from the hinge to a_u as radius. The intersection of these spheres (dark green) is a lens containing all possible placements of a_u. If the contact sphere (red) of a_p contains L there is a certain contact. If they are disjoint then contact is impossible.

Chapter 3

Applications of computational geometry

Many problems in structural biology are related to geometric problems, i.e. problems that can be stated in terms of two- or three-dimensional properties. Computational geometry is the field, within computer science, that deals with data structures and algorithms to solve such problems. One example is the computation of the volume of the union of a set of spherical atoms. Since atoms in a protein overlap, the problem is much harder than simply summing the volume of all spheres. This problem is relevant for finding the volume and even surface area of proteins and can be efficiently solved using so-called α-complexes [43] (explained in Section 3.2).

Since these problems are inherently geometric we have investigated if data structures from computational geometry can help reduce the running-time of protein structure representations and methods for scoring and analyzing structures. The first section discusses an improved representation of protein structures that enables efficient clash-detection during conformational search. The second section describes the work we did on protein packing quality and the final section outlines a data structure that we developed for topological analysis of protein structures.

3.1 Adjustable Chain Trees

As mentioned previously many representations of protein structures are based on the torsion angles of the backbone and side chain covalent bonds (see also Section 2.4). During conformational search a small number, k, of angles are changed, and the energy of the resulting conformation must be calculated. The direct approach is to update the positions of all n atoms accordingly (takes $\Theta(n)$-time) and then check all pairs of atoms to determine if the energy contribution from any pair has changed (takes $\Theta(n^2)$-time). Grid methods can be used to improve the efficiency of the energy calculation to $\Theta(n)$-time [44, 45]. Such conformational perturbations are the basis of most search methods and millions of moves are performed every second. Because detecting clashes and updating the energy is the most computationally requiring part of conformational search, improvements to the update time are extremely important.

Lotan et al. [44] describes a data structure called a *chain tree* which takes advantage of the chain structure in a protein to build a binary hierarchy of bounding volumes that is used to maintain a pairwise energy function. Given k torsion angle changes, updating the chain tree takes only $\mathcal{O}(k \log(n/k))$-time and calculating the energy change takes $\Theta(n^{1.5})$-time. In practice the chain

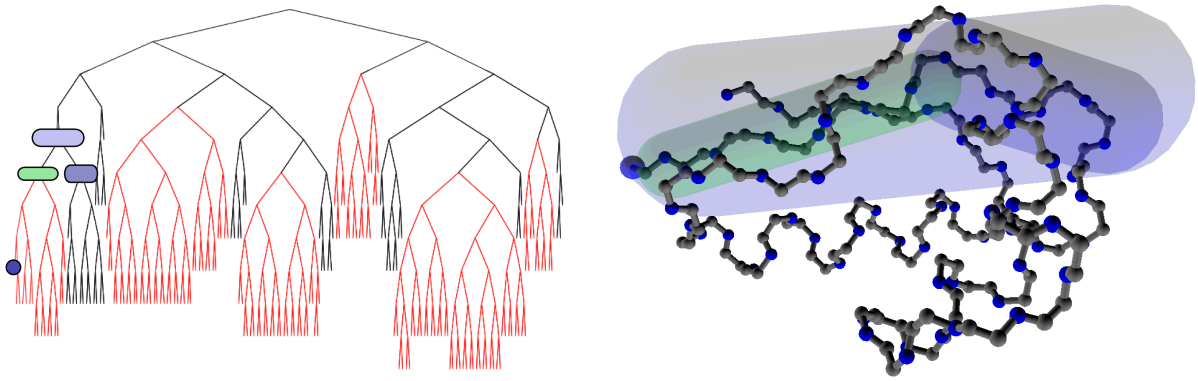


Figure 3.1: Left: Adjustable chain tree for the PDB-file 1CTF where three nodes and the N-terminal have been highlighted. The red sub-trees are the result of rearranging the chain tree such that secondary structure segments are exactly covered by a single node. **Right**: The bounding volumes corresponding to the highlighted nodes in the chain tree.

tree is shown to be faster than both the direct method and the grid method. Even though the chain tree is already a promising data structure we observed that not all relevant properties of protein structures were used to increase the speed. The paper "Adjustable Chain Trees for Proteins" (included in Section 5.5) therefore investigates how an improved version of the chain tree can be developed. The asymptotic running time stays unchanged but the adjustable chain tree is shown to be significantly faster when updating a protein structure.

As mentioned previously the first step in PSP is often to predict secondary structures (helices and strands) and 'lock' their local structure such that the torsion angles rarely change. In helical segments for example, the backbone torsion angles (ϕ, ψ) are set to values near $(-60°, -30°)$, which results in the characteristic helical shape. Helices and strands, therefore, become rigid and nearly straight segments. The efficiency of the chain tree is heavily dependent on how tight the bounding volumes fit the underlying chain. Since helices and strands are elongated rigid segments, the adjustable chain tree rearranged the hierarchy of bounding volumes such that each secondary structure segment had a node in the tree which exactly contains it (red sub-trees in Figure 3.1). Because the torsion angles within secondary structure segments rarely change, these bounding volumes hardly ever need to be updated and more computational time could be spent making them tight-fitting. The rearrangements might make the tree unbalanced so the tree was rebalanced in such a way that subtrees of helices and strands were unchanged.

One additional observation helped make the adjustable chain tree more efficient: Ω-torsion angles are generally fixed at 180° or 0°. The lowest levels of the adjustable chain tree was therefore rearranged to collect adjacent C_α-C-N-C_α atoms under one node in the tree. This means that the Ω-torsion angle was locked and bounding volumes were tightened as described above.

The original paper on chain trees used oriented bounding boxes (OBBs) and rectangular swept spheres (RSSs) as bounding volumes (see Figure 3.2). Both types of bounding volumes have good tightness of fit but finding the smallest volume and performing collision detection is computationally demanding. It was therefore investigated if a simpler type of volume with faster overlap checks could improve the update time of the adjustable chain tree. Since helices and extended backbones are shaped like long cylinders, a cylindrical volume seemed appropriate. Line swept spheres (LSSs)

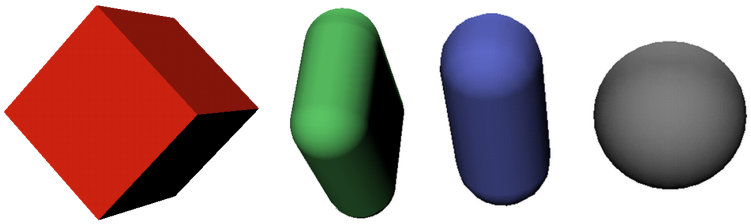


Figure 3.2: The different types of tested bounding volumes. From left, an oriented bounding box, a rectangular swept sphere (the Minkowski sum of flat 3D rectangle and a sphere), a line swept sphere (Minkowski sum of a 3D line segment and a sphere – also called a capped cylinder) and an ordinary sphere.

can be made at least as tight-fitting as a cylinder and has a much simpler overlap check. Both standard chain trees and adjustable chain trees were used to determine which type of volume (OBB, RSS, LSS or regular spheres) was best suited for bounding volume hierarchies in proteins. These experiments are described in the paper "Bounding Volumes for Proteins: A Comparative Study" (included in Section 5.6).

The test set contained a set of 12 proteins with varying lengths and secondary structure compositions. We formulated a cost-function indicating the computational cost of performing a rotation around a covalent bond by changing a torsion angle. The CPU-time of performing a rotation was also recorded. Both measures indicated that the locking of secondary structures and the locking of peptide bonds resulted in significant speed-ups. The two combined halved the average update-time of performing a rotation.

In the comparison of bounding volumes, the adjustable chain tree was three times faster with LSSs than with the RSSs used in the original paper by Lotan et al. Spheres had an extremely fast overlap check, but because they fit an elongated protein chain very poorly they ended up being slightly slower than the line swept spheres. By replacing a standard chain tree using RSSs with an adjustable chain tree using LSSs the average speed of performing a rotation tripled.

3.2 Packing quality

Given a set of points, P, in 2D, the Delaunay triangulation is the largest set of non-overlapping triangles with corners in P such that the circumcircle of any triangle contains no point from P in its interior. The α-complex can be interpreted as the subset of triangles in the Delaunay triangulation whose circumcircle has a radius less than some $\alpha \in \mathbb{R}^+$ (see Figure 3.3 for an example). Both Delaunay triangulations and α-complexes can be generalized to 3D using tetrahedra and spheres instead of triangles and circles. The 3D generalization of the Delaunay triangulation is called a *Delaunay tessellation*.

Delaunay tessellations and their dual, the Voronoi diagram, have been used for analyzing protein structures in a number of ways. One example is the *RosettaHoles* method for analyzing packing quality and finding packing defects using the cells of the Voronoi diagram [46, 47]. The problem is that protein structures that are generated using computational models tend to have small empty pockets in their interior that are not observed in experimentally determined native structures of

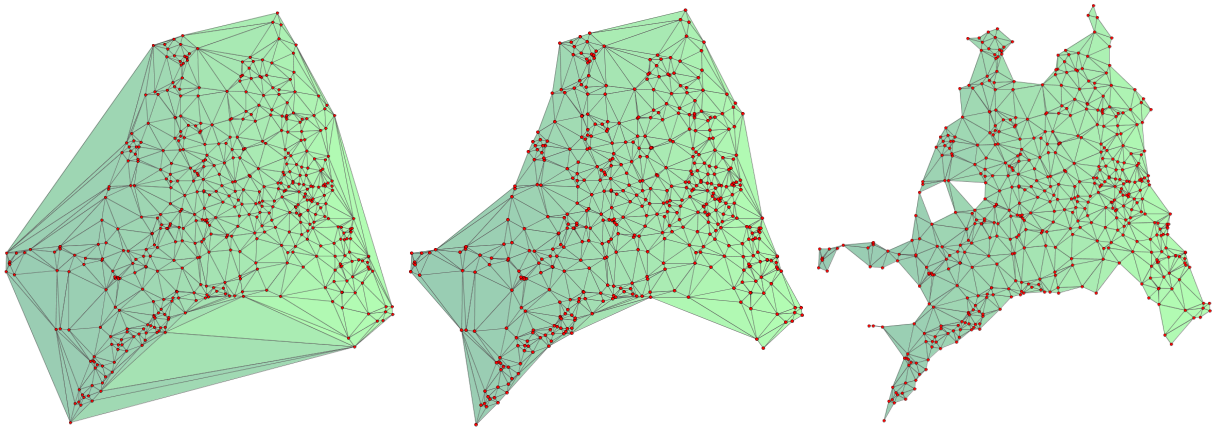


Figure 3.3: A set of points (500 random road intersections in Copenhagen) with the Delaunay triangulation and two α-complexes having $\alpha = 10$km and $\alpha = 1.8$km respectively.

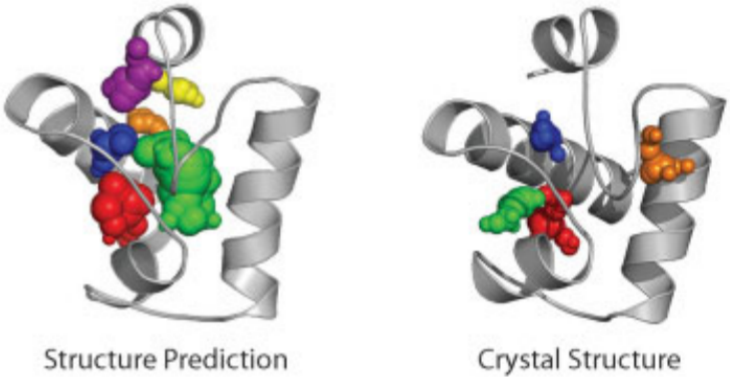


Figure 3.4: The difference between packing defects in a computationally generated structure prediction (left) and a high resolution crystal structures (right). The RMSD difference between the two structures is very small, but the packing defects are significant. The image is from [46].

high quality (see Figure 3.4). These pockets are not easily detected using any of the typical terms in scoring functions that only consider pairs of atoms. The distribution of distances from atom i to its neighbors might look reasonable even though there is a pocket just next to i. The pockets, however, are easily detected by analyzing the tetrahedra in a Delaunay tessellation because their circumspheres by definition are empty. The colored pockets in Figure 3.4 are unions of such circumspheres that have been shrunk slightly.

RosettaHoles assigns a packing score to each atom in the protein structure. This packing score is calculated by using the area of a series of sphere-shells intersected with the Voronoi cell (see left side of Figure 3.5) as inputs to a support vector machine (SVM). Determining the Voronoi cells and the area of intersected sphere-shells is computationally demanding. Therefore, when RosettaHoles is included in the Rosetta energy function, the conformational search slows down significantly. The paper "Protein Packing Quality using Delaunay Complexes" (included in Section 5.7) investigates how results comparable to those of RosettaHoles can be computed faster.

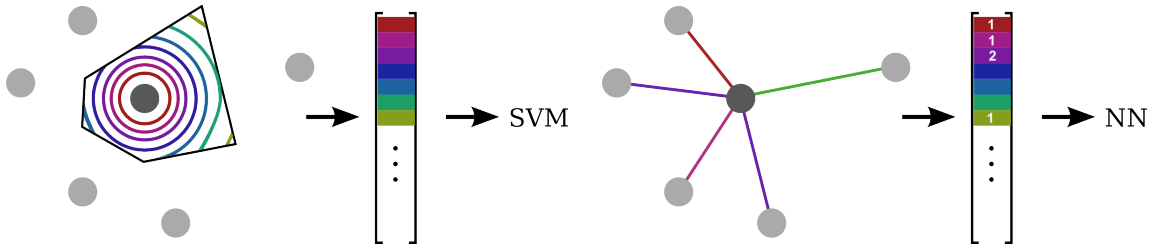


Figure 3.5: Left: The geometric basis for assigning packing scores to each atom in RosettaHoles is the area of concentric shells intersected with the Voronoi cell. **Right**: Our packing cost captures roughly the same information but uses the edge-lengths of Delaunay edges.

Instead of a support vector machine a neural network was used to assign a packing cost to each atom. The packing cost of the entire protein was the average of the packing cost of atoms. The input for a particular atom was a histogram of edge-lengths for all the edges in the Delaunay tessellation that are adjacent to the atom (see right side of Figure 3.5). The neural network was trained to distinguish atoms in well-packed structures, represented by structures solved with X-ray resolution less than 1.61Å, from atoms in poorly packed ones (represented by X-ray structures with resolution larger than 2.24Å).

The construction of the Delaunay tessellation is an incremental algorithm based on tetrahedron-walks and flips, but with a few improvements that take advantage of the protein structure. First, a big tetrahedron is placed around the entire point set. Each atom center is then inserted one at a time with the following steps.

- The tetrahedron, τ, containing the point is located by walking through adjacent tetrahedra, starting at a tetrahedron adjacent to the last inserted point.
- τ is split into four new tetrahedra
- Configurations of tetrahedra that violate the empty-sphere criterion are flipped until the structure becomes a Delaunay tessellation again.

Such insertion algorithms often randomize the order of points to guarantee a certain expected running-time[1]. The proposed method, however, adds points in the order they appear along the protein backbone. Since the tetrahedron-walk is always started from a tetrahedron adjacent to the last inserted point, it never walks very far and the time spent on this part is therefore minimized.

It was concluded that our method could successfully discriminate well-packed structures from poorly packed ones as shown in Figure 3.6. The packing cost furthermore correlated well with the packing quality score from the RosettaHoles2 method. The conclusion was that the neural network indeed characterized packing flaws and not some other structural difference between high- and low-resolution protein structures.

The computational speed of our method increased only linearly as the number of atoms increased (see Figure 3.7). The computational time of the RosettaHoles2 method, in contrast, had a nearly quadratic increase. Furthermore, our method was 3 to 4 times faster with any given input.

Our proposed method is fast and well-suited for scoring protein structures, but additional work

[1]See e.g. [48] for a good derivation of $\mathcal{O}(n \log n)$ expected time in 2D.

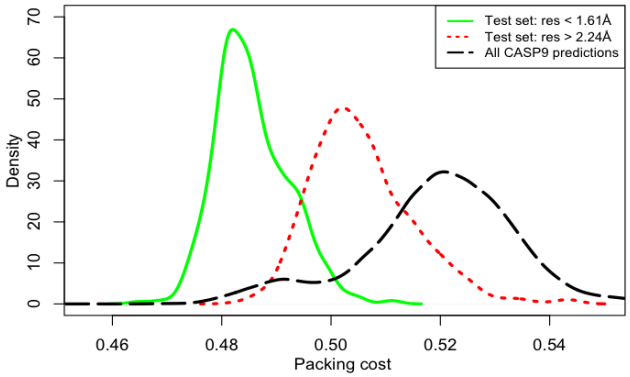


Figure 3.6: Distributions of packing costs for well-packed structures (green), poorly packed structures (red) and structures from the CASP9 experiment (black).

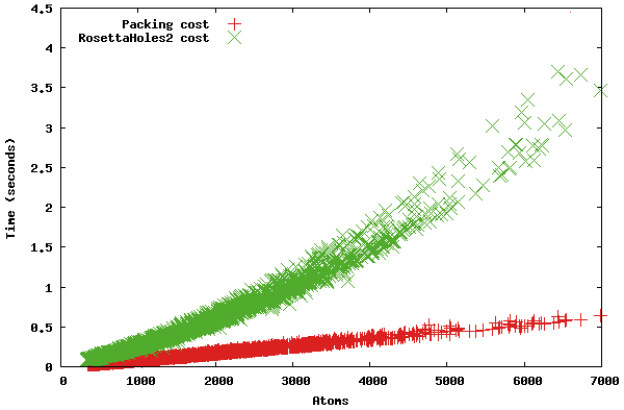


Figure 3.7: The computational time of our method (red) and RosettaHoles2 (green) as a function of protein size.

is needed before it can be integrated into an energy function. The use of neural networks and the discrete changes in the Delaunay triangulation as points move makes it difficult to express the derivative wrt. atom positions.

3.3 Characterizing the topology of a point set

While working on the packing cost, we tested if the geometric properties of empty holes, extracted from an α-complex, could be used to characterize packing quality. These experiments motivated a different type of problem within computational geometry.

Given a set of three-dimensional points, P, and the α-complex for some value of α, a hole can be detected by finding a topological *void*. A void is a set of tetrahedra not included in the α-complex but isolated from the exterior of the convex hull by triangles in the α-complex (see Figure 3.8). The size and number of voids depends on the chosen α-value. To avoid choosing one specific value, the entire evolution of the α-complex is considered as α increases from 0 to ∞. As α increases triangles are added, voids form, split up into smaller voids and finally disappear as

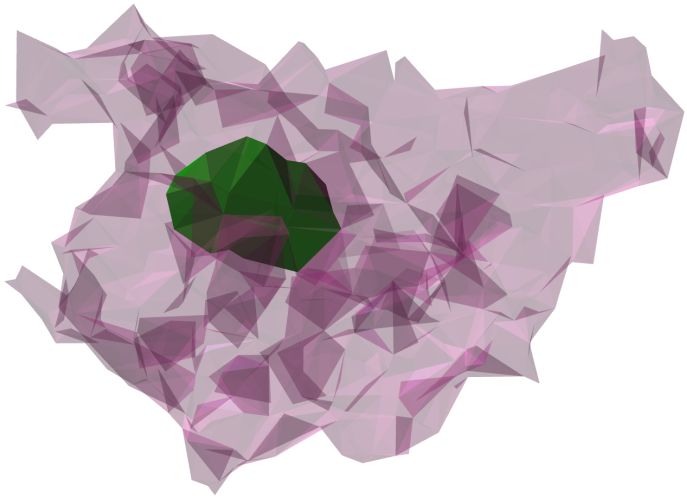


Figure 3.8: A topological void (green) within an α-complex where $\alpha = 2.8$Å. The point set corresponds to the atom-coordinates of the human ARNT protein (PDB-id: 1X0O).

tetrahedra are added. A void formed at an early α-value and disappearing at a late α-value is said to be a *persistent void*. There are very efficient computational methods for determining how persistent a void is [49, 50] but the practical problem considered here was how to efficiently represent and determine the tetrahedra contained within any void during the evolution of the α-complex.

The evolution of an α-complex can be represented as a *filtration*, which is a list of simplices (points, edges, triangles and tetrahedra) specifying the order in which each simplex is added to the α-complex. Some of the triangles (positive triangles) will create voids and each tetrahedron will fill a void. The *void tree* data structure, described in our paper "Visualizing and Representing the Evolution of Topological Features" (included in Section 5.8), is a new way of representing the parts of the evolution that concerns voids. By considering the entire empty space a void, any positive triangle will split an existing void in two, and every tetrahedron will fill a void. This observation was used to represent the evolution of voids as a binary tree where internal nodes corresponded to positive triangles and leaves corresponded to tetrahedra. Furthermore, children of a node were ordered such that the most persistent child was the right child. It was shown that a void tree could be constructed in $\mathcal{O}(n\alpha(n))$, where n is the number of simplices and $\alpha(n)$ refers to the extremely slowly growing inverse Ackerman function. This is just as efficient as the original calculation of persistence [50]. A void tree can also be constructed in 2D with positive edges and triangles instead of positive triangles and tetrahedra. Figure 3.9 gives an example of a 2D filtration and the corresponding void tree in 2D.

Void trees hold no more information than filtrations where the persistence of each feature has been calculated. Therefore, possible applications stem from representing the topological evolution as an ordered binary tree. For instance, the problem of collecting all tetrahedra contained in a void can be solved by locating the corresponding node in the tree, and collecting all the leaves in its sub-tree. Furthermore, the binary tree can prove useful in various applications as a visualization. The edit-distance between ordered binary trees [51] can describe how topologically similar two arbitrary point sets are by finding the edit-distance between their void trees. With this approach

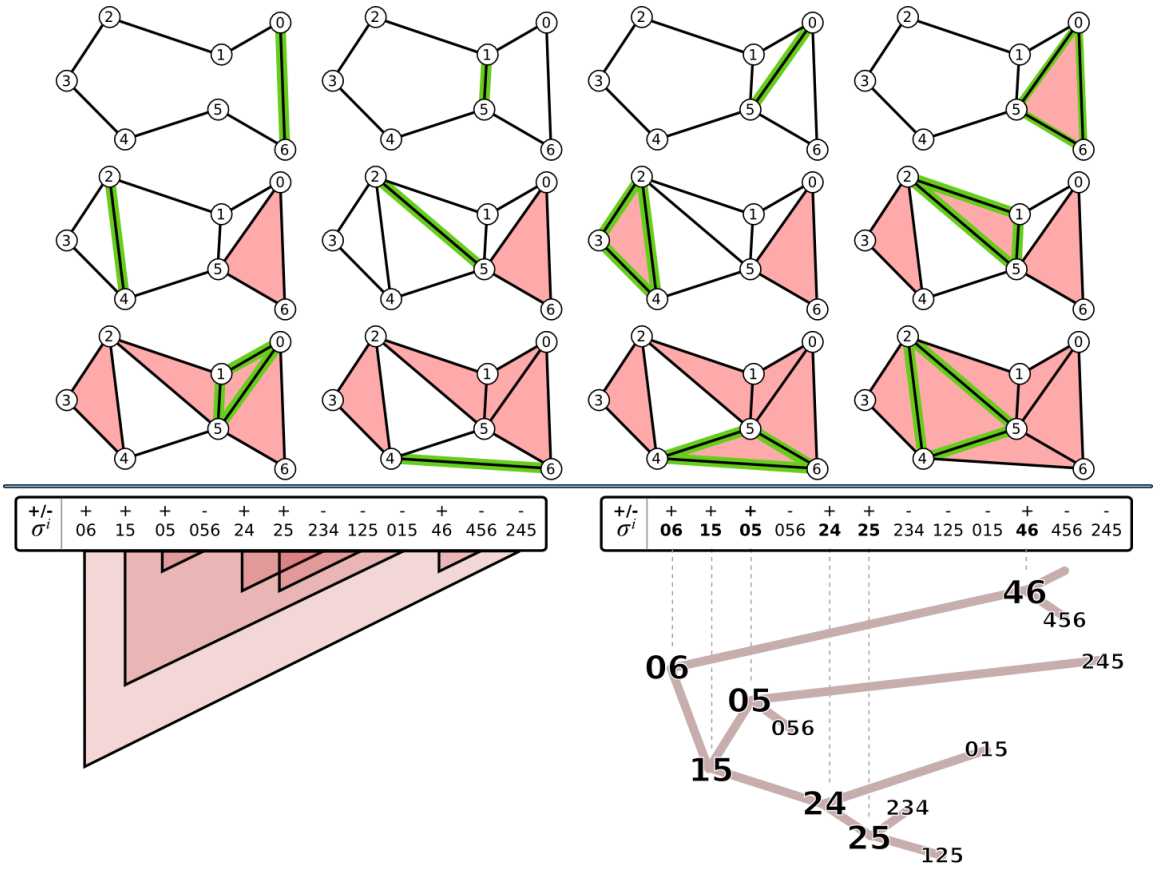


Figure 3.9: Top: An example of a 2D filtration. **Bottom left**: The original visualization of persistence. For example, edge 15 is positive because it creates a void. This void is filled by the triangle 015. The size of the triangle indicates the persistence of this void. **Bottom right**: The void tree drawn side-ways. The edge 06 splits the entire plane into two voids, the inside (lower branch) and outside (upper branch) of the shape.

there is no need for matching the points in the two sets or even for the sets to be of the same size.

We are currently working on extending the concept of void trees by building hierarchical representations for all topological features of 3D simplicial complexes. The evolution of connected components can be described using *component trees*, which are similar to dendrograms constructed with agglomerative single-linkage hierarchical clustering [52, 53]. The final type of feature, the so-called tunnels, can not be described using trees. It is possible that the evolution of tunnels can be represented as a directed acyclic graph but we have not found a good method to construct such a representation yet.

The observation that component trees are similar to hierarchical clustering implies that a void tree can be regarded as 'clustering of holes'. Since hierarchical clustering is widely used, the void tree might prove valuable as a supplementary method for data-mining as it conveniently characterizes any absence of data points in a point sets interior. The practical down-side is that the α-complex is required in the construction of the void tree which is complicated to compute in higher dimensions

or in non-Euclidean spaces. To counter these problems, it would be interesting to investigate void trees extracted from so-called witness complexes [54], which provide an alternative to the α-complexes.

Chapter 4

Conclusion and future directions

PSP remains a difficult problem and each advance within the field comes slowly and requires large engineering efforts. It is difficult to say with certainty how the problem will one day be solved, but it is widely acknowledged that there are currently two bottlenecks in the field of molecular modelling: Better methods are needed to efficiently sample the search space of proteins, and energy functions that better approximate the Gibbs free energy must be developed [55, 56]. This thesis has primarily addressed the former.

There are many possible future directions that spring from this study, but two seem particularly promising to me. First, I hope to complete the work on using sheets in the conformational search as outlined in Chapter 2. There is still much work in representing the structure of sheets, doing efficient loop-closure and thinking of intelligent ways to search the energy landscape.

The second direction is to develop new energy functions based on Delaunay tessellations. In recent papers the focus has been on packing of atoms, but the Delaunay tessellation holds more information about spatial relations that could be relevant in describing biophysical properties of macromolecules. The first challenge will be to efficiently maintain the changes in the tessellation that occur as atoms move during the conformational search. There has been some work done on kinetic Delaunay tessellations, but including knowledge of how atoms move in proteins could make these data structures even faster and convince more researchers to employ them. The second challenge will be to improve energy functions by designing and incorporating meaningful three- and four-body interactions in addition to the existing pair-wise interactions.

Chapter 5

Papers

5.1 Predicting Dihedral Angle Probability Distributions for Protein Coil Residues From Primary Sequence using Neural Networks

The following 8 pages contains the published version of our paper "Predicting Dihedral Angles of Coil-structure Residues from the Primary Sequence using Neural Networks. G. Helles and R. Fonseca. BMC Bioinformatics 2009, 10:338" [57].

BMC Bioinformatics

Research article

Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks

Glennie Helles* and Rasmus Fonseca

Address: University of Copenhagen, Department of Computer Science, Universitetsparken 1, 2100 Copenhagen, Denmark

Email: Glennie Helles* - glennie@diku.dk; Rasmus Fonseca - rfonseca@diku.dk

* Corresponding author

Published: 16 October 2009

BMC Bioinformatics 2009, **10**:338 doi:10.1186/1471-2105-10-338

Received: 6 April 2009

Accepted: 16 October 2009

This article is available from: http://www.biomedcentral.com/1471-2105/10/338

Abstract

Background: Predicting the three-dimensional structure of a protein from its amino acid sequence is currently one of the most challenging problems in bioinformatics. The internal structure of helices and sheets is highly recurrent and help reduce the search space significantly. However, random coil segments make up nearly 40% of proteins and they do not have any apparent recurrent patterns, which complicates overall prediction accuracy of protein structure prediction methods. Luckily, previous work has indicated that coil segments are in fact not completely random in structure and flanking residues do seem to have a significant influence on the dihedral angles adopted by the individual amino acids in coil segments. In this work we attempt to predict a probability distribution of these dihedral angles based on the flanking residues. While attempts to predict dihedral angles of coil segments have been done previously, none have, to our knowledge, presented comparable results for the probability distribution of dihedral angles.

Results: In this paper we develop an artificial neural network that uses an input-window of amino acids to predict a dihedral angle probability distribution for the middle residue in the input-window. The trained neural network shows a significant improvement (4-68%) in predicting the most probable bin (covering a 30° × 30° area of the dihedral angle space) for all amino acids in the data set compared to baseline statistics. An accuracy comparable to that of secondary structure prediction (≈ 80%) is achieved by observing the 20 bins with highest output values.

Conclusion: Many different protein structure prediction methods exist and each uses different tools and auxiliary predictions to help determine the native structure. In this work the sequence is used to predict local context dependent dihedral angle propensities in coil-regions. This predicted distribution can potentially improve tertiary structure prediction methods that are based on sampling the backbone dihedral angles of individual amino acids. The predicted distribution may also help predict local structure fragments used in fragment assembly methods.

Background

The primary sequence of a protein is believed to define the three-dimensional (tertiary) structure of the protein and many attempts at predicting the tertiary structure from primary sequence has been made (see for instance [1] for an overview of the CASP VIII experiment).

The main reasons that predicting protein structure from sequence alone is so difficult, is that the possible ways the

amino acids can twist and turn with respect to each other are enormous. However, large parts of most proteins are arranged in secondary structures like helices and sheets, in which the dihedral angles of the amino acids lie within fairly limited areas as can be observed in Ramachandran plots [2-4]. Fortunately, predicting secondary structures can be done quite accurately [5-8], and since roughly 60% of amino acids in most proteins are arranged in these secondary structures [9], the number of possible amino acid conformations is dramatically decreased by this information. When attempting to predict the tertiary structure of proteins, the intermediate step of determining the secondary structure is thus typically performed.

It is important to note, though, that even if all helices and sheets in a protein have been predicted correctly, finding the complete tertiary structure is still a problem of daunting size. First of all, the dihedral angles of residues in secondary structures are still relatively flexible. Secondly, the dihedral angles of residues in coil segments are very flexible and they do not show any simple recurrent pattern like those in helices and sheets.

By inspecting the Ramachandran plot of large sets of proteins it is evident that although coil residues generally populate a much larger and more diverse area than helical and strand residues, certain dihedral angles are nearly never encountered. Steric overlap between atoms in the side chains of adjacent resides are believed to be responsible for this, indicating that flanking residues have a significant effect on the dihedral angles of a given residue, but exactly how big an effect remains unclear. Erman et al. [10] showed that, although the exact structure cannot be unequivocally determined by flanking residues, the structure is largely affected by these. On the other hand, Kabsch et al. [11] have shown that identical sequences of five residues in different proteins may still adopt different structures, which means that the exact dihedral angles of a residue cannot be determined strictly from the local environment.

Predicting the exact dihedral angle area of a coil residue based only on flanking residues thus appears to be infeasible, but we may still be able to predict the most probable dihedral angle areas. When residues are predicted as helix or strand residues, we are also provided with a most probable dihedral angle area. Using this information, de novo protein structure prediction methods allow us to direct the search to areas of the dihedral angle space where we are most likely to find the correct conformation.

A predicted probability distribution can therefore be used as either an alternative to fragment assembly, which, although it has improved tertiary structure prediction significantly, suffers from the fact that it relies heavily on known structures, or as a tool that can help improve the prediction success of the local fragment predictions used by fragment assembly algorithms [12-14]. A significant amount of work has already been done in predicting these local fragments [15-20], but as noted in [15], dihedral angle propensities are used in this prediction process and a neural network prediction of dihedral angle preferences could likely aide the prediction.

In this work we attempt to predict a dihedral angle probability distribution for coil regions that can be used by tertiary structure prediction algorithms to sample the conformational space more efficiently. Using a dihedral angle probability distribution does not restrict the dihedral angle space, but rather suggests a frequency to which we should search different areas of the dihedral angle space in order to increase the probability of finding the right dihedral angles for an amino acid.

Neural networks are well known for their ability to learn and extract patterns from massive amounts of data, so we have chosen to use this method to generate probability distributions. Neural networks have also previously played an important role in predicting secondary structures [5,7,8].

To our knowledge, predicting dihedral angle probability distributions of coil residues only have not previously been done. However, both Kuang et al. and Zimmermann and Hansmann have attempted to predict dihedral angle areas of coil residues and we have used them for inspiration. Both groups divide the Ramachandran plot into three main areas representing approximately 90-100% of the dihedral angle space and then they try to predict in which of the three areas the dihedral angles a coil residues would be in. Kuang et al. used both a neural network and support vector machine but they reported only marginal differences in performance for the two different prediction methods and ended up with an overall prediction accuracy of 77% for the 25% PDBSelect data set (February 2001 version) [21]. Zimmermann and Hansmann used support vector machines to create three classifiers; one for each part of the Ramachandran plot. They report a higher accuracy of between 81.7% and 93.3% for the 50% PDBSelect data set [22]. We wish to emphasize that unlike Kuang et al. and Zimmermann and Hansmann we are not concerned with predicting a single predefined area containing the correct dihedral angles. Instead, we attempt to predict a probability distribution that will yield the most probable dihedral angle area for a given residue in a given sequence. Hamelryck et al. developed a hidden markov model to predict probability distributions of dihedral angles [23], but their analysis was not limited to coil-regions and comparable results were not presented.

In the *Methods* section the method for constructing and training the neural network is described. Section *Discussion* presents and discusses the results, and section *Conclusions* draws the final conclusion.

Results and Discussion

Two methods of evaluating the neural network are used. The first method measures the accuracy using only a single bin. The second include several bins and describe the accuracy of the predicted dihedral angle distribution.

Lower bound on prediction accuracy

While a probability distribution can be constructed based on the results, the neural network is trained to predict a single bin. Table 1 shows the prediction accuracy for each type of amino acid. The prediction accuracy is the percentage of coil-residues for which the neural network had highest output in the bin corresponding to the correct dihedral angle. In order to determine the significance of the results presented, it is useful to compare them with the probability of guessing the right bin based on the distribution of dihedral angles in the data set. Simply guessing at the most populated bin for coil residues would yield a successful guess at a rate of:

$$G = \frac{R_{\text{most}}}{R_{\text{total}}} \quad (1)$$

Where R_{most} is the number of residues in the most populated bin and R_{total} is the total number of residues in the data set. We may think of G as a lower bound on the prediction accuracy. This lower bound can be tightened by analyzing plots specific to each type of amino acid. For instance Figure 1B shows the probability distribution for threonines that has been calculated using this equation. Lower bounds for the neural networks prediction accuracy, specific to each type of amino acid, $G^{\text{AA-type}}$, can thus be determined.

As can be seen from Table 1 the trained neural network yield better accuracies than $G^{\text{AA-type}}$ and the number of correctly predicted bins are improved for all types of amino acids. Improvements of more than 50% compared to guessing are observed for 7 out of the 20 residues. The largest improvement observed is for threonine where the correct bin is predicted by the neural network 68% more frequently than guessing at the most populated bin. Predicting dihedral angles for valine shows the smallest improvement of only 4%.

Interestingly, the neural network appears to perform better on hydrophilic residues, as 7 of the 9 hydrophilic residues are the ones that showed improvements of more than 50%. Only the hydrophilic residues, arginine and glutamic acid, showed improvements of less than 50% (but still >40%). In contrast, prediction for most hydrophobic residues showed improvements of less than 35%. This distinction between hydrophobic and hydrophilic residues may of course be mere coincidence, but it does seem to indicate that hydrophilic residues are much more controlled by their local environment than the hydrophobic residues, which are not as easily influenced. This is

Table 1: Improvements in prediction accuracy.

AA-type	Property	$G^{\text{AA-type}}$	NN Prediction	Improvement
arg	I	9.7%	13.6%	40.2%
asn	I	8.3%	13.3%	60.2%
asp	I	8.2%	13.7%	67.1%
gln	I	8.7%	13.3%	52.9%
glu	I	10.6%	15.1%	42.5%
his	I	7.7%	12.0%	55.8%
lys	I	9.7%	14.9%	53.6%
ser	I	10.9%	16.5%	51.4%
thr	I	9.1%	15.3%	68.1%
gly	-	15.0%	16.2%	8.0%
ala	O	12.4%	17.3%	39.5%
cys	O	10.5%	14.0%	33.3%
ile	O	14.3%	15.3%	7.0%
leu	O	12.5%	16.0%	28.0%
met	O	9.8%	12.3%	25.5%
phe	O	10.4%	12.6%	21.2%
pro	O	21.4%	27.0%	26.2%
trp	O	13.5%	15.2%	12.6%
tyr	O	9.4%	12.0%	27.7%
val	O	13.7%	14.2%	3.6%

Prediction accuracy of the neural network is compared to a lower bound derived from a purely statistical analysis of the data set. 'O' and 'I' in the "property" column denotes hydrophobic and hydrophilic residues respectively.

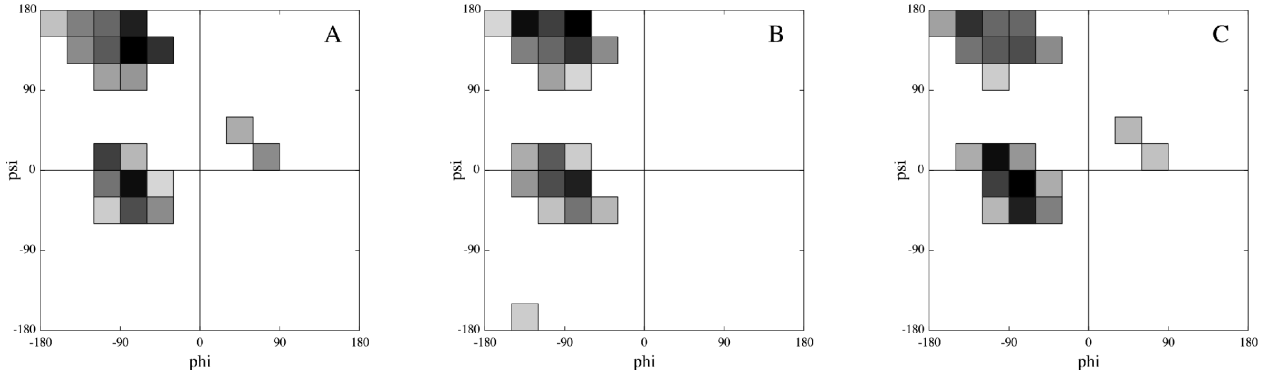


Figure 1
Bin distribution. The plot to the left (A) shows the distribution of the 20 most populated 30° × 30° bins for all coil residues in the training set. The plot in the middle (B) shows the distribution for just threonines in the training set, and the plot to the right (C) shows the predicted bins for threonine in the sequence Glu-Leu-Asp-Thr-Glu-Asp-Ala taken from a randomly chosen protein in the data set. The neighboring residues are used by the neural network to suggest a different distribution to yield a higher success rate. The darker the color of the bin the more likely it is that the angle set is within this bin.

completely in keeping with the assumption that hydrophobic packing is the driving force in protein folding.

Guessing based only on the distributions observed in Ramachandran plots would yield a success rate of roughly 8-15% for all residues except proline that has an unusual high accuracy of 21%. Even large improvements of 4-68% will thus only bring the overall prediction accuracy up to roughly 12-27%, which is of course insufficient for reliable coil prediction. However, Figure 2 shows the prediction accuracy of the neural network compared to simple statistics based prediction when observing more than one bin. On average, neural network based prediction performs better as long as we look at an area that includes less than 55 bins. The highest gain in prediction accuracy com-

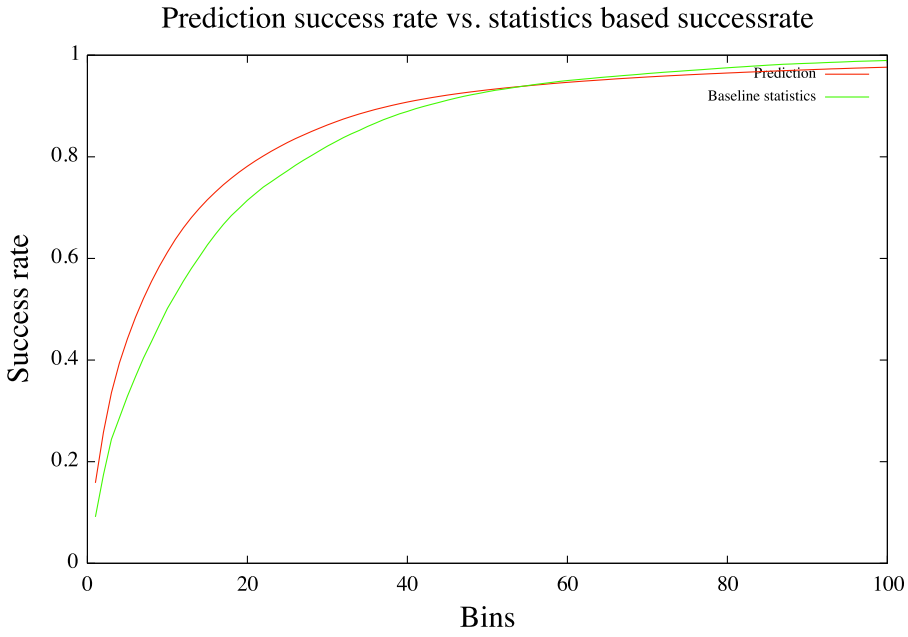


Figure 2
NN prediction vs. baseline statistics. Prediction vs. baseline statistics.

pared to baseline statistics is achieved when we look at the 8 bins with highest output values.

Accuracy of probability distribution

The above comparison with the lower bound indicates that the neural network is learning more than just baseline statistics, and that the flanking residues do in fact play a role for the local structure. However, our goal is not to predict a single bin, but rather to create a probability distribution for an area of the Ramachandran plot that will give us as high a prediction accuracy for any given sequence. With a prediction accuracy of ≈ 80% for secondary structures most tertiary structure prediction algorithms incorporates secondary structure predictions as a way to limit the search space. As already mentioned, residues in secondary structures do in fact span a rather large dihedral angle subspace, and so the question is whether we are able to obtain a similar accuracy for an equally sized area.

The increase in success rate for each included bin is depicted for each type of amino acid in Figure 3. As can be seen the average prediction accuracy for all residues is just under 80% (78%) within the 20 top scoring bins. For proline, which appears to be the easiest to predict, an accuracy of 80% is achieved within the dihedral angle area covered by the top eight scoring bins whereas glycine, which is by far the most difficult to predict, need to span an area covering 40 bins in order to achieve an ≈ 80% accuracy.

Comparison

Both Kuang et al. [21] and Zimmermann & Hansmann [22], who attempted to predict dihedral angle areas of coil residues, divided the Ramachandran plot into three areas. Their smallest area (area A in [21], area H in [22]) has roughly the same size as 21 of our 30° × 30° bins. The second smallest area (area B in [21], area E in [22]) has an area corresponding to 25 of our bins and the largest area (referred to as area E/G in [21] and area O in [22]) corresponds to more than 80 of our bins - in fact in [22] area O simply takes up the remaining part of the Ramachandran plot.

Kuang et al. report an overall prediction accuracy of 77% and we thus achieve a higher accuracy per area ratio. Zimmermann et al. report an accuracy of 82.1% for area H, 81.7% accuracy for area E and 93.3% accuracy for their outlier area O. Again all areas are larger than ours and their improved accuracy over Kuang et al. are likely due to their use of the 50% PDBSelect data set, rather than the 25% PDBSelect data set used by both [21] and us. Generally, sequences with 50% or more sequence identity can

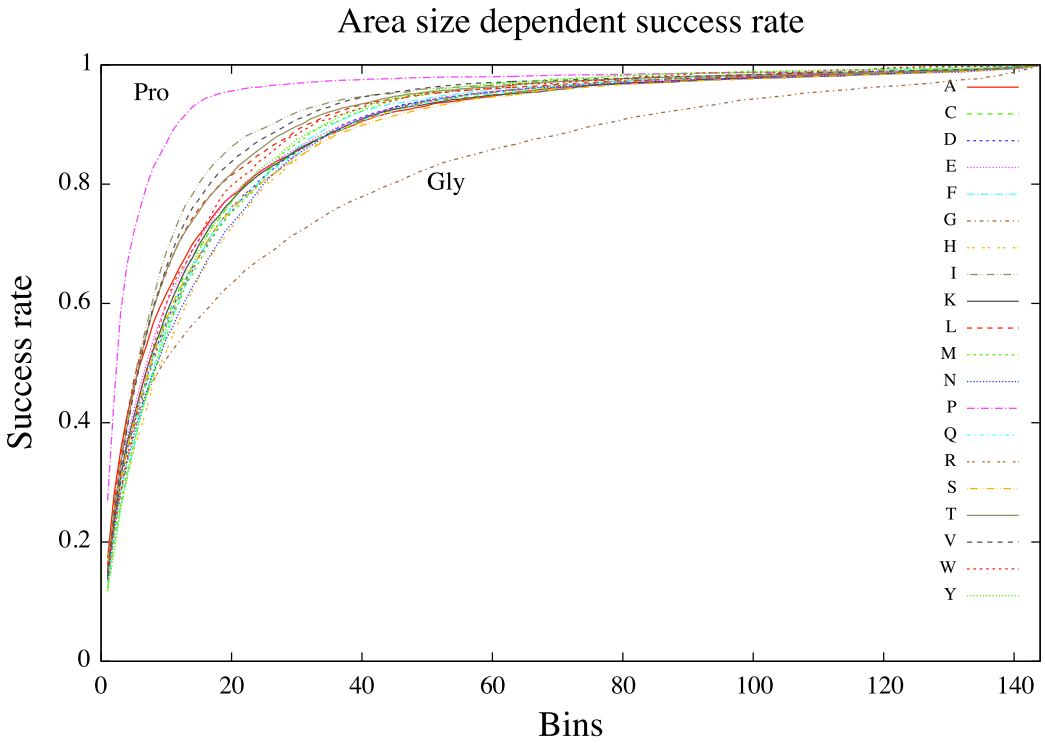


Figure 3
Success rates. Area size dependent success rate. Each bin represents a 30° × 30° area of the Ramachandran plot.

be assumed to adopt the same three-dimensional structure whereas structures with only 25% cannot [24]. The classification algorithm used by Zimmermann and Hansmann thus have a natural advantage as their data set is not as diverse. Comparing the accuracy per area ratio, however, is not completely fair, since we are essentially trying to solve two different problems. Large areas like those in [21,22] are well suited for some tasks, but for limiting the search space in de novo protein structure prediction, we deem smaller bins more useful.

Figure 1C shows an example of the area predicted for threonine in a randomly chosen sequence from the data set. Figure 1A and 1B show plots drawn directly for all residues and only threonine in the data set respectively. The neural network clearly learns a different distribution based on the surrounding amino acids that will yield a better prediction accuracy for that specific sequence.

Future work

An extension of the neural network, that may improve the results, would be to distribute the bins differently but still keep them relatively small. Preferred areas of turns [25] could be represented explicitly with bins or the optimal size of bins could be examined in more detail. Another possibility for future work is to assign higher target value to bins near the target (Φ, Ψ) point during training of the neural network. In this work the bin containing the target point is assigned 0.9 and all others 0.1. Due to the flexibility of the backbone the real point may easily be in one of the neighboring bins, so these could be assigned a target value of e.g. 0.5 during training. This could possibly help the neural network to generalize better.

Another extension is to train 20 individual neural networks, one for each amino acid. We have here chosen the network that had the best overall prediction accuracy for all of the amino acids, but from our experiments it is clear that individual residues often peaked at different times during the training procedure. We thus expect that the results we have reported here can be improved by training a network for each amino acid type.

Conclusion

Our work shows that artificial neural networks can predict a probability distribution of dihedral angle areas for residues in a protein fast and better than simple statistics. For a dihedral angle area corresponding in size to those associated with helices and sheets that can be predicted with a ≈ 80% accuracy we achieve comparable results with a 78% accuracy. To our knowledge, results from attempts to predict probability distributions has not previously been reported, but it could prove very useful in guiding search algorithms for de novo protein structure prediction toward the most probable areas of the search space, much in the same way that predicted secondary structures do.

Methods

A fully connected feed-forward neural network was constructed and used to predict a 30° × 30° dihedral angle bin corresponding to the (Φ, Ψ)-coordinates of the *target residue*.

We used the May 2008 25% PDBSelect data set http://bioinfo.tg.fh-giessen.de/pdbselect/recent.pdb_select25, which consists of 3881 chains (553016 residues) with less than 25% sequence identity (20 chains were omitted in our data set because we were unable to obtain information about secondary structures with DSSP). In this experiment we are only interested in predicting probability distributions for coil residues, so we used information about secondary structures from the DSSP-algorithm [26]. A reduction from the eight groups of DSSP (3_{10}-helix, α-helix, π-helix, β-bridges, β-sheets, turns and bends) was performed by classifying all residues that are either β-bridges, β-sheets, 3_{10}-helices or α-helices as 'secondary structure' and the rest as 'coil'. This reduction corresponds to method A described in [27]. The neural network was trained on 'coil'-residues alone, though 'secondary structure' residues were often present in some part of the input window. Residues at the end of chains where either Φ or Ψ values are undefined were omitted.

The data set was split randomly in two equally sized sets, PDBSelect25_A and PDBSelect25_B. PDBSelect25_A was used to determine an appropriate network configuration and PDBSelect25_B was then used to obtain the prediction results reported in this work.

The input to the neural network was a window that spanned W residues of the amino-acid sequence with the target residue in the center. A number of experiments were run to determine the neural network configuration that would yield the highest prediction accuracy. Prediction accuracy was calculated as the percentage of coil residues from a validation set for which the neural network could predict the correct bin. Window sizes, W, of 5, 7 and 9 were used with various numbers of hidden neurons, H. Generally speaking, more hidden neurons are needed for larger input windows, but rather than experimenting with a fixed number of hidden neurons we simply kept increasing the number of hidden neurons with 50 until performance showed no improvements. Based on these experiments we settled on a window size of $W = 7$ and a neural network with $H = 100$ hidden neurons in a single hidden layer. We emphasize that while we have made experiments with many different architectures, we have not systematically verified that the neural network is optimal for this task, but as all architectures achieved almost

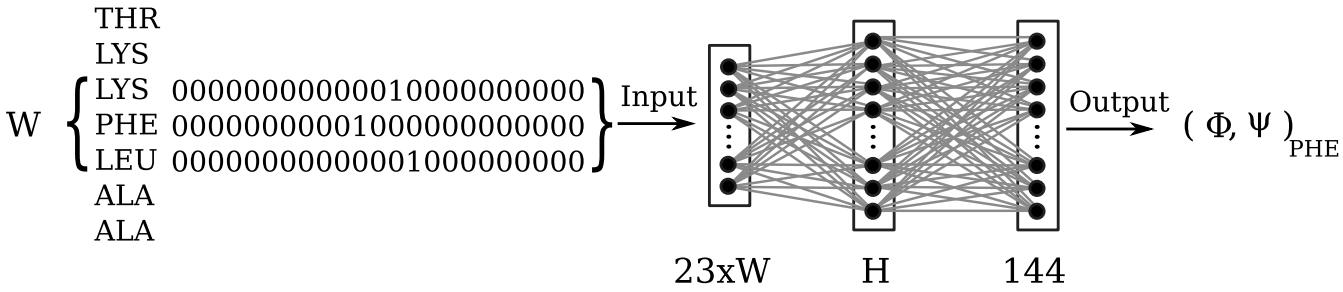


Figure 4
Network configuration. Workflow of the prediction. The residues in the input-window is encoded and used as input to the neural network that passes values through a hidden layer. The predicted (Φ, Ψ)-area can be read from the output-layer.

the same prediction accuracy we feel confident that changing the architecture is unlikely to change the prediction accuracy in any major way.

The neural network was designed so it had 23 input neurons per residue in the input window. One neuron was used to specify if the residue was part of a secondary structure (helix or strand), one was used to specify if the residue was part of a coil, one neuron was used to indicate if the input was a dummy (outside a chain or an unknown amino acid) and the 20 remaining input neurons were used to uniquely identify each of the 20 amino acid types. Neither the dummy nor the secondary structure input neurons are ever set to 1 for the middle residue. Using 20 input neurons to represent the residue is common and the procedure roughly corresponded to the one used by [7] to predict secondary structure. Figure 4 shows an overview of the neural network design.

The 144 neurons in the output-layer each correspond to a 30° × 30° area of the Ramachandran-plot. It was estimated that this size would be sufficiently small to be of use and sufficiently big to ensure that uncertainties in dihedral angles would not prevent the neural network from being able to learn. The expected output value of a certain area was 0.9 if the Φ and Ψ-angles of the middle residue of the input window fell within the boundaries of this bin, and 0.1 otherwise. We used 0.9 and 0.1 rather than 1 and 0 to ensure faster convergence with the standard logistic sigmoid activation function that was used in all layers. We used the standard sigmoid function because it is fast and because we are essentially only interested in finding the highest output signals and not the output value per se. The neural network was trained using standard back-propagation with learning momentum. The learning parameters of the back-propagation algorithm was set to $\gamma = 0.05$ (learning rate) and $\alpha = 0.1$ (learning momentum).

For the initial experiments with different neural network configurations we split the PDBSelect25$_A$ data set randomly into five subsets. Four of them was used for training one for validation. Training was then carried out for 10.000 epochs with the weights updated after each training example. The highest prediction accuracy was achieved within the first 1000 epochs in all experiments. After 1000 epochs the prediction accuracy showed the slow decline for the unknown validation set and the slow increase in the training set that is the typical sign of overfitting.

Once we settled on a neural network configuration we trained and validated the network on the PDBSelect25$_B$ data set. Like the PDBSelect25$_A$ data set, the PDBSelect25$_B$ data set split randomly into five subsets where four were used for training and one was used for validation. Since we previously achieved the highest prediction accuracy within the first 1000 epochs, we cut the training time down to 5000 epochs, but otherwise the hyper-parameters were identical to the ones already described. We ran a traditional 5-fold cross validation to ensure that the PDBSelect25$_B$ data set had not been split inappropriately. As is evident from Table 2, the neural network was able to predict the correct 30° × 30° bin approximately 16% of the times regardless of the way the data set was split.

Table 2: 5-fold cross validation results.

	Prediction accuracy$_{singlebin}$
Split A	16.2%
Split B	15.0%
Split C	15.9%
Split D	15.9%
Split E	15.8%
Avg	15.7%

The data set was randomized and split into five separate sets and we carried out a 5-fold cross validation. The results from each fold is listed here along with the average.

Authors' contributions

GH conceived of the study and carried out implementation of the neural network. RF has helped design the study and been responsible for data acquisition. Both authors have been involved in the literature study and both have drafted, read and approved the manuscript.

Acknowledgements

We would like to thank Pawel Winter for his help in drafting this manuscript as well as the anonymous reviewers that gave helpful technical suggestions for training neural networks.

References

1. Ben-David M, Noivirt-Brik O, Paz A, Prilusky J, Sussman JL, Levy Y: **Assessment of CASP8 structure predictions for template free targets.** Proteins: Structure, Function, and Bioinformatics; 2009.
2. Ramachandran GN, Ramakrishnan C, Sasisekharan V: **Stereochemistry of polypeptide chain configurations.** *J Mol Biol* **7:**95-99.
3. Barlow DJ, Thornton JM: **Helix geometry in proteins.** *J Mol Biol* 1988, **201(3):**601-19.
4. Sibanda BL, Blundell TL, Thornton JM: **Conformation of beta-hairpins in protein structures. A systematic classification with applications to modelling by homology, electron density fitting and protein engineering.** *J Mol Biol* 1989, **206(4):**759-77.
5. Jones D: **Protein secondary structure prediction based on position-specific scoring matrices.** *Journal of Molecular Biology* 1999, **292(2):**195-202.
6. Chu W, Ghahramani Z, Wild DL: **A graphical model for protein secondary structure prediction.** In *ICML '04: Proceedings of the twenty-first international conference on Machine learning* New York, NY, USA: ACM; 2004:21.
7. Qian N, Sejnowski TJ: **Predicting the secondary structure of globular proteins using neural network models.** *J Mol Biol* 1988, **202(4):**865-884.
8. Rost B, Sander C: **Prediction of Protein Secondary Structure at Better than 70% Accuracy.** *Journal of Molecular Biology* :584-599.
9. Creighton T: *Proteins. Structures and Molecular Properties* Freeman, New York; 1993.
10. Keskin O, Yuret D, Gursoy A, Turkay M, Erman B: **Relationships Between Amino Acid Sequence and Backbone Torsion Angle Preferences.** *Proteins: Structure, Function, and Bioinformatics* 2004, **55:**992-998.
11. Kabsch W, Sander C: **On the use of sequence homologies to predict protein structure: Identical pentapeptides can have completely different conformations.** *Proceedings of the National Academy of Sciences of the United States of America* 1984, **81(4):**1075-1078.
12. Klepeis JL, Floudas CA: **ASTRO-FOLD: A Combinatorial and Global Optimization Framework for Ab Initio Prediction of Three-Dimensional Structures of Proteins from the Amino Acid Sequence.** *Biophysical Journal* 2003, **85(4):**2119-2146.
13. Zhang Y, Wu S, Skolnick J: **Ab initio modeling of small proteins by iterative TASSER simulations.** *BMC Biology* :17+.
14. Martin Paluszewski PW: **Protein Decoy Generation Using Branch and Bound with Efficient Bounding.** *Algorithms in Bioinformatics* 2008:382-393 [http://www.springerlink.com/content/9777ug388370g341].
15. Fernandez-Fuentes N, Oliva B, Fiser A: **A supersecondary structure library and search algorithm for modeling loops in protein structures.** *Nucl Acids Res* 2006, **34(7):**2085-2097.
16. Hunter CG, Subramaniam S: **Protein local structure prediction from sequence.** *Proteins: Structure, Function, and Genetics* 2003, **50(4):**572-579.
17. Fourrier L, Benros C, de Brevern AG: **Use of a structural alphabet for analysis of short loops connecting repetitive structures.** *BMC Bioinformatics* 2004, **5:**58.
18. Etchebest C, Benros C, Hazout S, Brevern AD: **A structural alphabet for local protein structures: improved prediction methods.** *Proteins: Structure, Function, and Bioinformatics* 2005, **59(4):**810-827.
19. Sander O, Sommer I, Lengauer T: **Local protein structure prediction using discriminative models.** *BMC Bioinformatics* 2006, **7:**14.
20. Katzman S, Barrett C, Thiltgen G, Karchin R, Karplus K: **Predict-2nd: a tool for generalized protein local structure prediction.** *Bioinformatics* 2008, **24:**2453-2459.
21. Kuang R, Leslie CS, Yang AS: **Protein backbone angle prediction with machine learning approaches.** *Bioinformatics* 2004, **20:**1612-1621.
22. Zimmermann O, Hansmann UHE: **Support vector machines for prediction of dihedral angle regions.** *Bioinformatics* 2006, **22:**3009-3015.
23. Boomsma W, Mardia KV, Taylor CC, Ferkinghoff-Borg J, Krogh A, Hamelryck T: **A generative, probabilistic model of local protein structure.** *Proceedings of the National Academy of Sciences* :8932-8937.
24. Rost B: **Twilight zone of protein sequence alignment.** *Protein Engineering* 1999, **12:**85-94.
25. Perskie L, Street T, Rose G: **Structures, Basins and Energies: A Deconstruction of the Protein Coil Library.** *Protein Science* 2008, **17(7):**1151-1161.
26. Kabsch W, Sander C: **Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features.** *Biopolymers* :2577-2637.
27. Cuff J, Barton G: **Evaluation and improvement of multiple sequence methods for protein secondary structure prediction.** *Proteins: Structure, Function and Genetics* 1999, **34(4):**508-19.

5.2 Protein Structure Prediction Using Bee Colony Optimization Metaheuristic

The following 14 pages contains the published version of our paper "Protein Structure Prediction using Bee Colony Optimization Metaheuristic. R. Fonseca, M. Paluszewski and P. Winter. Journal of Molecular Modelling and Algorithms, 2010" [58].

J Math Model Algor (2010) 9:181–194
DOI 10.1007/s10852-010-9125-1

Protein Structure Prediction Using Bee Colony Optimization Metaheuristic

Rasmus Fonseca · Martin Paluszewski · Pawel Winter

Received: 1 March 2009 / Accepted: 1 December 2009 / Published online: 19 February 2010

Abstract Predicting the native structure of proteins is one of the most challenging problems in molecular biology. The goal is to determine the three-dimensional structure from the one-dimensional amino acid sequence. *De novo* prediction algorithms seek to do this by developing a representation of the proteins structure, an energy potential and some optimization algorithm that finds the structure with minimal energy. Bee Colony Optimization (*BCO*) is a relatively new approach to solving optimization problems based on the foraging behaviour of bees. Several variants of BCO have been suggested in the literature. We have devised a new variant that unifies the existing and is much more flexible with respect to replacing the various elements of the BCO. In particular, this applies to the choice of the local search as well as the method for generating scout locations and performing the waggle dance. We apply our BCO method to generate good solutions to the protein structure prediction problem. The results show that BCO generally finds better solutions than simulated annealing which so far has been the metaheuristic of choice for this problem.

Keywords Protein structure prediction · Bee Colony Optimization · Metaheuristic

Partially supported by a grant from the Danish Research Counsil (51-00-0336).

R. Fonseca (✉) · P. Winter
Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
e-mail: rfonseca@diku.dk

P. Winter
e-mail: pawel@diku.dk

M. Paluszewski
The Bioinformatics Centre, University of Copenhagen, Copenhagen, Denmark
e-mail: palu@binf.ku.dk

1 Introduction

Proteins are the primary building blocks in all living organisms. They are made of amino acids bound together by peptide bonds. Depending on the sequence of amino acids, the proteins fold in three dimensions so that the Gibbs energy is minimized. The shape determines the function of the protein. *Protein structure prediction* (PSP) is the problem of predicting this three-dimensional structure from the amino acid sequence and is considered one of the most important open problems of theoretical molecular biology. The PSP problem has applications in medicine within areas like drug- and enzyme design [12].

PSP proves to be a very difficult optimization problem. Solving it exactly is only possible when using very simplified models. Use of heuristics is therefore necessary for more detailed models and energy functions. However, even in simplified scenarios, many computational problems arise. One of these problems is the belief that free energy landscapes tend to have many local minima [11].

Lately, several optimization heuristics inspired by bee colonies have been proposed. The two main approaches are the evolutionary algorithms and the foraging algorithms. The evolutionary approach was initially proposed in [1] and was based on the mating of bee drones with a queen bee. The foraging approach was proposed simultaneously in [17, 18] and [8, 9] and mimics the foraging behaviour of honey bees searching for and collecting nectar in a flower field. This heuristic, like real honey-bees, performs a wide search for good solutions and has a flexible method for allocating resources to intensify the local searches. This seems like a good strategy in the PSP to avoid getting stuck in the local minima of the energy landscape. Several names have been given to the foraging algorithm but here *Bee Colony Optimization* (BCO) is chosen.

Bahamish et al. [2] previously used the *Bees Algorithm* [17] to find the native state of the 5-residue peptide 'met-enkalphin' (PDB-ID: 1PLW) using a full resolution torsion angle-based representation. We apply the BCO metaheuristic to the PSP problem for real-sized proteins using a simplified representation. Good quality solutions, often called decoys, in terms of the RMSD and GDT similarity measures, are generated. These decoys can be used as starting solutions for more advanced methods. Since a coarser representation is used, real-sized protein structures can be attacked by the BCO metaheuristic. This is the first time a bee heuristic has been used to predict the structure of real-sized proteins (more than 50 residues). We do not claim to solve the PSP or even compete with state-of-the-art PSP algorithms like Rosetta [19] or I-Tasser [27]. However, the BCO metaheuristic has appealing properties. For instance, the scout bees make sure several local minima are searched, and the waggle dances intensify the search in promising areas. We believe this makes BCO suitable for the PSP.

In Section 2 the representation and the energy function of proteins are described. In Section 3 our adaptation of BCO is specified. Finally, experiments are described in Section 4 and discussed in Section 5.

2 Protein Model

The representation of proteins is important since it determines the size and conformation of the search-space. The following section describes our representation of the proteins structure.

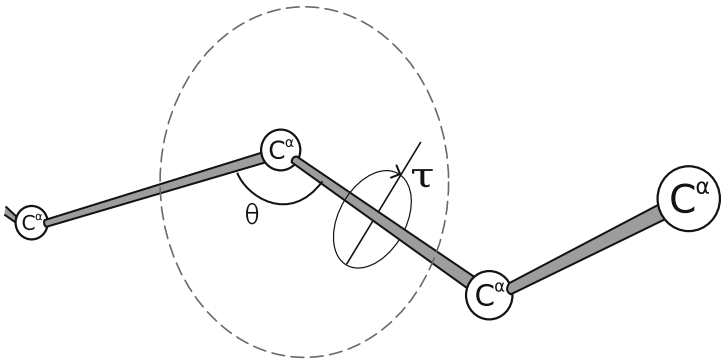


Fig. 1 C_α trace of backbone. Each amino acid is assigned two angles: θ and τ

2.1 Segment Representation

There are 20 different kinds of amino acids, each represented by a letter. The letter-sequence of amino acids is called the *primary structure* of the protein. Frequently occurring local structures of amino acids, such as helices and strands, are called *secondary structures* and the full description of the protein (i.e., 3D coordinates of all atoms) is called the *tertiary structure*.

When trying to determine the overall tertiary structure of a protein, sometimes the side chains and the atoms of the backbone are disregarded, and only the central carbon atom, C_α, of amino acids are represented. This leads to the C_α-trace representation of proteins illustrated in Fig. 1. The entire protein structure can be represented by assigning two angles to each amino acid, θ and τ.

Each amino acid of a protein can be classified as belonging to exactly one secondary structure. Here three classes of secondary structures are considered: helix, strand and coil. Helices and strands are distinguished by the unique geometrical layout of the C_α atoms in the tertiary structure (left part of Fig. 2) which is caused by a special pattern of hydrogen bonds. Coil is the class of amino acids that do not have a regular pattern of hydrogen bonds, and therefore has only few geometric constraints on the tertiary structure.

A sequence of C_α-atoms of the same secondary structure class is here called a *segment*. The secondary structure class of all amino acids is predicted and used as part

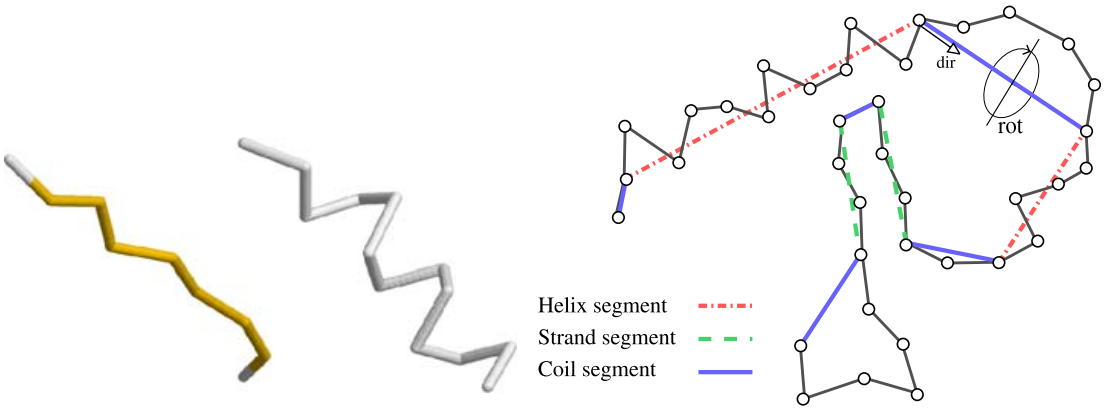


Fig. 2 *Left*: typical structures for strand and helix (generated using RasMol [20]). *Right*: segment representation of a protein

of our model. Segments can be considered as rigid rods that define the overall path of C_α-atoms belonging to the segment. Segments always have a start coordinate and a direction, and for helices and strands their end-coordinate can also be determined because of their constrained geometry. A segment is an abstract representation and does not explicitly contain the coordinates of internal C_α-atoms. The specification of how the C_α-atoms are arranged around the segment is called the *segment structure*. Together, all segments and segment structures constitute the *complete structure*, or simply the protein structure. The right part of Fig. 2 is an illustration of a complete structure using the segment representation.

The tertiary structure of any protein can be described by a complete structure. However, to discretize and reduce the conformational space of this model, the degree of freedom for segments and segment structures is reduced. Segments are therefore only allowed to have a discrete number d of predefined directions between the first and last C_α-atoms. Also, the number of possible segment structures is limited to s. The method used to determine the segment structures of helix, strand and coil-classes is described in Section 2.2. Obviously, the chance of being able to represent a complete structure similar to the native structure of the protein increases when more directions and segment structures are allowed, but this also increases the size of the conformational space.

The complete structure of a protein with m segments is represented by a pair of integers for each segment indicating the segment direction and segment structure.

$$(d_i, s_i), \qquad i = 1 \ldots m, \quad d_i \in \{1 \ldots d\}, \quad s_i \in \{1 \ldots s\}$$

2.2 Segment Structures

In this section it is described how the s allowed segment structures of a given segment are computed. This computation depends on the secondary structure class of the segment.

Helix and Strand Structures The most observed angle pair for an amino acid is $(\theta, \tau) = (91°, 49°)$ in helices and $(\theta, \tau) = (120°, 163°)$ in strands. Given a helix or strand segment, one segment structure having these angle properties is generated. Then the other $s - 1$ segment structures are generated by rotating the first structure uniformly around the axis going through the first and last C_α-atoms.

Coil Structures There are no simple geometric constraints that describe coil structures. However, experiments show that short sequences with similar amino acid sequences, so-called homologous sequences, often have similar tertiary structures [4]. Given a coil segment, the PDBSelect-25 dataset [3] is queried to find the $\sqrt{s}$ best matching structures. Each of these structures is rotated uniformly $\sqrt{s}$ times such that a total of s segment structures is obtained.

2.3 Energy

Determining a simple energy function for protein structures that is computationally fast and correlates well to native structures is still an open problem. Pseudo-energy functions are based on statistical analysis of large sets of proteins. These types of

energy functions are usually very fast but the quality of the minimal energy structures varies greatly.

A promising pseudo-energy function described in [16] is based on *Half-Sphere Exposure* (HSE) [5] and *Contact Numbers* (CN). This function requires very little computation and represents many of the crucial aspects of native structures (for instance side-chain surface exposure and residue burial). An important property of the HSE and CN measures is that they can both be predicted fairly accurately, so the energy of a structure can be calculated as the deviation from the predicted values.

For a given amino acid, the HSE is a pair of integers describing how many amino acids are contained in a half-sphere *above* the amino acid and how many are contained in the half-sphere *below* (See Fig. 3). The plane dividing the two half-spheres is specified by the position of the C_α-atom, A_i, and an $\overrightarrow{up}$-vector specific to the amino acid. The $\overrightarrow{up}$-vector can be defined in the following way.

$$\overrightarrow{up} = \overrightarrow{A_{i-1}A_i} + \overrightarrow{A_{i+1}A_i}$$

This $\overrightarrow{up}$ vector is undefined for the first and last amino of the protein, so for these only the contact number *CN* can be calculated. The CN for every amino acid is the number of amino acids contained in the *entire* sphere. The HSE and CN vectors specifying all the up/down numbers and contact number can be predicted from the primary structure alone using support vector regression [23, 25].

Let $\mathcal{P}$ denote the conformational space of a protein with n amino acids. Let $p \in \mathcal{P}$. The total energy $Q(p)$ is defined as the sum of the individual energy contributions $Q_p(i)$ from each amino acid i, i.e.,

$$Q(p) = \sum_{i=1}^{n} Q_p(i) \tag{1}$$

where

$$Q_p(i) = \begin{cases} \Delta CN(i)^2 & \text{if } i \text{ is the first amino acid of a segment.}^1 \\ \Delta HD(i)^2 + \Delta HU(i)^2 & \text{otherwise} \end{cases}$$

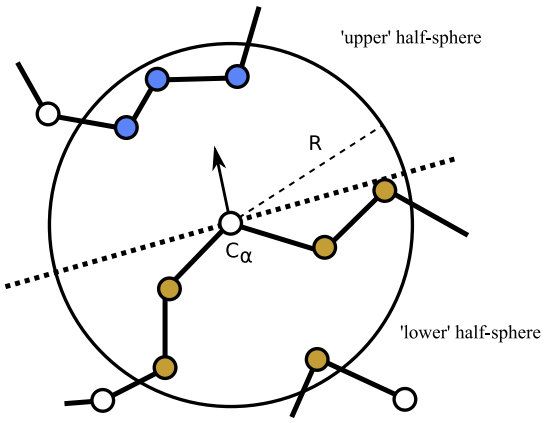


Fig. 3 Half-sphere exposure for an amino acid. The up/down pair is (3, 5). The contact number is 8

and

- $\Delta CN(i)$ is the difference between the actual contact number of the i-th amino acid and the desired (i.e., predicted).
- $\Delta HD(i)$ is the difference between the down half sphere exposure number of the i-th amino acid and the desired down half sphere exposure number.
- $\Delta HU(i)$ is the similar difference for the up half sphere exposure.

A radius of the HSE-sphere around 13 Å is known to give a good prediction quality [24] and it seems to capture both local and non-local contacts. The optimal radius has yet to be determined, both in terms of predictability and information content.

Since many amino acids are hydrophobic, globular proteins fold into tight spheric conformations. An HSE based energy function is not enough to ensure this behaviour, so the mean squared distance of the amino acids from the protein center, the *radius of gyration* (Rg), is introduced. The center is defined as the average position of C_α atoms in the structure. Rg can be predicted from the number of amino acids n of the protein [22]:

$$Rg_{\text{pred}} = 2.2n^{0.38} \tag{2}$$

This prediction is often accurate for globular proteins. Infinite energy is therefore assigned to structures having $Rg > 1.2 \cdot Rg_{\text{pred}}$.

A structure is said to be clashing if the distance between two C_α-atoms is less than 3.5 Å. A clashing structure is also assigned infinite energy.

3 Bee Colony Optimization

In nature, a foraging bee can be said to be in one of three states: A scout bee, a worker bee or an onlooker. Scout bees fly around a flower field at random and when a flowerbed is found, they return to the hive and perform a waggle dance. The dance indicates the estimated amount of nectar, direction and distance to the flowerbed. Onlooker-bees present in the hive watch different waggle dances, choose one and fly to the selected flowerbeds to collect nectar. Worker bees act like scout bees except that when they have performed the waggle dance they return to their old flowerbed to retrieve more nectar instead of flying out at random. A bee usually chooses to become a worker bee when the chosen flowerbed has a very high concentration of nectar.

In our adaptation of the BCO metaheuristic, each bee corresponds to a specific complete solution, and the nectar amount corresponds to an objective value in the energy landscape. Sending out scout bees corresponds to finding a random feasible solution and sending out onlookers corresponds to performing a local search iteration on some existing solution. The onlookers choose a solution for local search based on the objective value of scout and worker-solutions in previous iterations.

[1]The reason why CN instead of HSE is used for the first amino acid of each segment is that it was necessary for the Branch and Bound algorithm described in [15, 16]. In order to compare solutions found here with those in [16] the same energy function is preserved.

This method is largely the *Bees Algorithm* proposed in [18]. In a non-changing solution space, the fitness of a solution does not deplete in the same way a real life flowerbed depletes of nectar. Exhaustion is therefore forced when a worker-solution can not be improved. This idea is somewhat similar to the idea of pruning parts of the search space as described in [14]. The process of exhausting a search around a worker-solution is proposed as part of the *Artificial Bee Colony* algorithm described in [9]. Our adaptation of the BCO metaheuristic is a synthesis of these two approaches.

Algorithm 1 BCO pseudocode

```
0  saved ← ∅
1  pop ← ScoutStrategy(W + S)
3  while Stopping criterion is not met
4      for each p ∈ pop do
5          onlookers[p] ← OnlookerStrategy(Cost(p), ∑_{p'} Cost(p'), O)
6          p ← NeighborhoodSearch(p, onlookers[p])
8          if Cost(p) has not improved for Exhaust iterations then
9              saved ← saved ∪ {p}
10             p ← ScoutStrategy(1)
11     newScouts ← ScoutStrategy(S)
11     Replace the S solutions in pop that has the worst costs with newScouts
10 return The best solution—either from pop or from saved
```

Here S, W and O is the number of scout, worker and onlooker bees, respectively. OnlookerStrategy is the strategy for assigning onlookers and NeighborhoodSearch(p, o) is the neighborhood strategy for performing o iterations of local search around a solution, p. ScoutStrategy(s) is a method for generating a set of s random solutions. The S worst solutions in the population are always scout-solutions which means that the W best ones are worker-solutions. It is not specifically indicated how new solutions should be generated using ScoutStrategy, how onlookers should be assigned or how local search in NeighborhoodSearch should be performed. Depending on the nature of a problem each of these three methods can be designed to fit the problem. New solutions can be generated with genetic algorithms by using mutation and crossover in ScoutStrategy, and any of the numerous existing local search heuristics can be used as NeighborhoodSearch. The basic procedure however is to let ScoutStrategy generate a random solution and set NeighborhoodSearch to perform hill-climbing. Using this basic procedure it is observed that:

$$\mathrm{BCO}(S, W, O, \mathit{Exhaust} = \infty) = \mathrm{BA}(S, W, O)$$

$$\mathrm{BCO}(S = 0, W, O, \mathit{Exhaust}) = \mathrm{ABC}(W, O, \mathit{Exhaust})$$

Where BA is the Bees Algorithm [18] and ABC the Artificial Bee Colony algorithm [8]. The above representation of the foraging bees optimization paradigm is more generally applicable than the ones presented in [18] and [8] since both are special instances of BCO.

3.1 Bee Colony Optimization Applied to PSP

Algorithm 1 can be used for any optimization problem where OnlookerStrategy, NeighborhoodSearch and ScoutStrategy are defined, so to utilize BCO for PSP these three methods have to be specified. The energy function $Q(p)$ (Eq. 1) is used as cost-function.

*Scout Search Strategy (*ScoutStrategy*)* To find a random feasible solution, a depth first search is used to determine the direction d_i and structure s_i of each segment i. At each level in the depth first search, a random ordering of direction and structure is tried so the same solution is not generated every time.

*Onlooker Choosing Strategy (*OnlookerStrategy*)* A number of onlookers are assigned to a solution j among the scouts and workers based on the costs of the population and the amount of onlookers O. Onlookers are assigned probabilistically based on a fitness given by:

$$fitness(j) = \frac{1/\text{Cost}(j)}{\sum_{j'} 1/\text{Cost}(j')}$$

The OnlookerStrategy, however ensures that only a total of O onlookers are assigned in each iteration.

*Onlooker Neighborhood Strategy (*NeighborhoodSearch*)* Any local search could be utilized as neighborhood strategy but a simple hill-climbing strategy is chosen. A neighbor solution is generated by changing directions d_i of two randomly chosen segments and the segment structure s_i of four randomly chosen segments. If the cost improves the new solution is accepted.

4 Experiments and Results

Two sets of experiments are performed, one on a simple model and one on a flexible model.

The *simple model* allows $d = 12$ basic directions defined by the face-centered lattice and $s = 8$ rotations for each segment: $d_i \in \{1..12\}, s_i \in \{1..8\}$. This choice of d and s ensures a reasonable flexibility of the structure, but also makes the search-space sufficiently small to allow the EBBA algorithm [16] to find an optimal solution within 48 h.

The *flexible model* allows $d = 73$ basic directions defined by a combination of the face-centered, body-centered and simple cubic lattices and $s = 16$ rotations for each segment: $d_i \in \{1..73\}, s_i \in \{1..16\}$. This choice of d and s is made to ensure that the model is much more flexible than the simple model. The purpose of creating a flexible model is to see if a metaheuristic can find solutions with lower energy than the optimal energy found in the simple model.

The energy function, $Q(p)$, is the same for both models. The distance measure RMSD(p) and the Global Distance Test (GDT) measure [26] can be compared for structures belonging to both of these models as well as structures obtained using completely different models and methods. GDT$_c(p)$ is calculated as the largest set

of amino acids in some structure p that can be superposed on to the native structure such that the distance of each amino acid in the set is less than c from the amino acid position in the native structure. GDT(p) is defined as the average of $GDT_1(p)$, $GDT_2(p)$, $GDT_4(p)$ and $GDT_8(p)$.

The first set of experiments uses BCO, EBBA and a simulated annealing algorithm to find good decoys using the simple model for six proteins. These six proteins have previously been used for benchmarks in the literature [6, 16, 21]. The input to the optimization algorithms is a secondary structure assignment, the HSE-vectors and the radius of gyration. For each protein these values are obtained using prediction tools. Based on the amino acid sequence, the secondary structure is predicted using PSIPRED [13] and HSE-vectors using LAKI [24]. For better comparison of energy levels, the HSE predictions from [16], which were done using LAKI [24], were used. The radius of gyration is predicted using Eq. 2. The six benchmark proteins used here also exist in PDB, so there is a slight chance that the training sets for PSIPRED and LAKI contain some of these proteins. However, the prediction quality of the six benchmark proteins is close to what should be expected. We therefore do not consider it to be a problem that the benchmark proteins exist in PDB.

In order to be able to compare the results from BCO with the results obtained using EBBA [16], BCO is run for 48 h. Due to this large run-time extensive parameter-optimization has not been considered. Considering the runtimes for generating a scout and finding a valid neighbor-solution, we estimate that using $S = 10$ scouts, $W = 10$ workers, $O = 100$ onlookers and $Exhaust = 5$ is appropriate. We also wish to compare BCO to a commonly used metaheuristic within bioinformatics, so a Simulated Annealing (SA) algorithm is implemented. New solutions are generated using the ScoutStrategy method described in Section *Scout Search Strategy (*ScoutStrategy*)* and the local search is based on the NeighborhoodSearch method also described in Section *Onlooker Neighborhood Strategy (*NeighborhoodSearch*)*. The only difference in the NeighborhoodSearch method is that the SA algorithm accepts solutions that have ΔQ worse energy with the probability $p = e^{\frac{-\Delta Q}{T}}$. We tested different starting temperatures from 0.5 to 16 and measured the ratio of accepted changes out of all the valid changes. Johnson et al. [7] studied the simulated annealing heuristic and found that this ratio should be between 20% and 90%. We therefore set the starting temperature to $T = 1$ because the resulting acceptance ratio was around 50%. A linear annealing schedule was chosen such that the temperature falls to $T = 0$ when SA terminates. This eliminates the need to decide a final temperature.

Ideally SA converges on optimal solutions if allowed to cool down sufficiently slow. To improve the quality of solutions obtained by SA, restarts are often suggested in the literature. We settled on a relatively low number of restarts (ten restarts, each taking 4.8 h) as more restarts would make SA a special variant of BCO.

Figure 4 shows a single run of EBBA and five runs of BCO. The time-cost curves show that BCO is very stable and finds the optimal value of $Q(p)$ almost as fast as EBBA. Similar experiments were performed for SA but it always converged on a suboptimal value after roughly 5–6 h.

The second round of experiments are performed using BCO and SA, but this time on the flexible model. Using the flexible model EBBA can no longer find the optimal value in 48 h. In addition to the six previously benchmarked proteins we also attempt to predict good structures for two somewhat bigger targets from

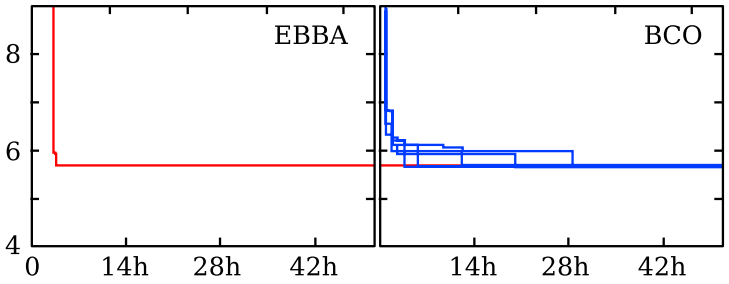


Fig. 4 Best observed energy vs. time for EBBA and BCO using the simple model

CASP7 [10]. We have intentionally chosen a pair that proved to be hard to predict by CASP7 participants. Most successful CASP7 methods were homology-based. Since our algorithm is not using homology modelling, it should be compared with PSP methods for proteins with no good templates in PDB. For the two CASP proteins, the newer and more accurate HSE prediction server HSEpred [23] was used instead of LAKI.

To make BCO and SA collect decoys, we stored the 1,000 best structures with respect to $Q(p)$ encountered during a search. For comparison and evaluation of the model and prediction quality, the second round of experiments was also done using the exact secondary structures, exact HSE-vectors and exact radius of gyration

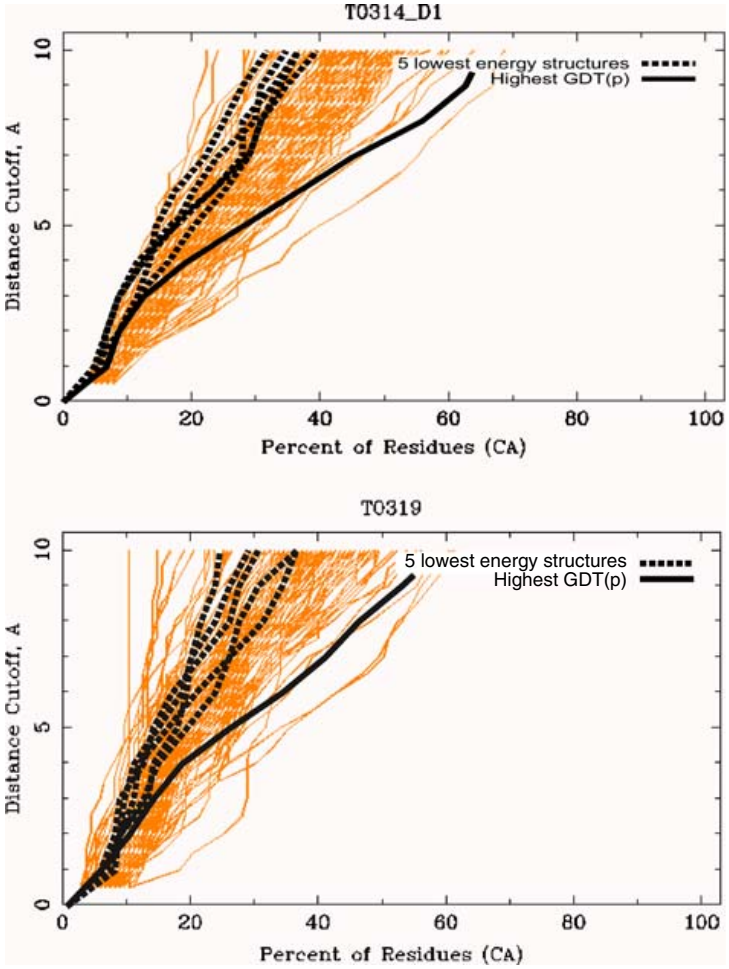


Fig. 5 GDT analysis plot for the proteins 2HG6 (*top*) and 2J6A (*bottom*). The x-axis shows the offset, c, and the y-axis shows $\mathrm{GDT}_c(p)$. The *orange curves* are structures from all the predictors at CASP7. The *blue curves* are the five structures with lowest energy $Q(p)$ generated by BCO. The *green curve* is the structure $p^\dagger$ with highest $\mathrm{GDT}(p)$. *Both blue and green curves* are found using predictions of secondary structure, HSE vectors and radius of gyration

obtained from the native structures of the proteins. These structures cannot be considered solved *de novo*. All computations were performed on a 3.4 GHz Intel Xeon with 2 GB RAM.

Table 1 summarizes the results of the runs from BCO and SA using the flexible model, EBBA using the simple model as well as the results from CASP7. p^* is the protein structure encountered during a search for which the energy $Q(p)$ is lowest. For BCO, SA and EBBA the energy function is identical. $p^\dagger$ is the protein structure—among the 1,000 saved decoys—for which the similarity (GDT(p)) is highest.

Figure 5 show GDT analysis plots for the proteins 2HG6 and 2J6A. The GDT analysis is a type of plot used at CASP to indicate how good the $\text{GDT}_c(p)$ is, using different distance cutoff values c. A curve lying to the far right correspond to a conformation near the native structure. All the orange curves in the figures correspond to structures generated by the participants at CASP7. The blue curves all correspond to structures generated by BCO using predicted secondary structure, predicted HSE-vectors and predicted radius of gyration. The green curves correspond to $p^\dagger$ structures with highest GDT(p).

5 Discussion and Conclusion

The results of BCO and SA compared to those achieved at CASP7 are shown for the proteins 2HG6 and 2J6A in Table 1. It can be seen that the HSE-based energy function does not completely identify the best structure since GDT(p^*) is relatively low for BCO and SA. This can also be observed from the GDT analysis in Fig. 5. The five structures with lowest energy result in curves that are slightly worse than the average at CASP7. If, however, a more advanced energy function is applied to the 1,000 generated structures then $p^\dagger$ can possibly be identified. Using GDT(p) as quality measure this would rank the structures obtained by BCO as 30-th out of 132 for 2HG6 and 17-th of 132 for 2J6A at CASP7. The curves illustrating $p^\dagger$ in the GDT analysis are even more promising as $\text{GDT}_{10}(p^\dagger)$ are among the highest for both 2HG6 and 2J6A. This indicates that the model and the BCO heuristic are good at finding structures where the 'overall' conformation is close to the native. This is a notable achievement for an energy function that is primarily based on predicted HSE numbers and radius of gyration.

When comparing BCO to SA, the focus should be on the values of $Q(p^*)$ since both algorithms optimize the energy. For all the problems, except 2GB1 exact, BCO achieves a lower value of $Q(p^*)$ which indicates that BCO is superior to SA on these types of problems. The average values of $Q(p^*)$ for the six smaller proteins are illustrated in Table 2. For these proteins BCO finds values of $Q(p^*)$ that, on average, are 5% better than those found by SA. It is worth noting that Monte-Carlo based algorithms like SA usually are the metaheuristics of choice for PSP.

When looking at the results for 1FC2 (exact) and 1ENH (exact), it is clear that they differ from the other rows. The lowest energy observed is less than 3 for both runs which is considerably lower than for the other runs. It is remarkable that the corresponding very low energy structures are native-like. This supports the hypothesis that HSE, secondary structure and radius of gyration contain enough information to identify the native structure of the protein. There are two possible

Table 1 Results from Bee Colony Optimization (BCO), Simulated Annealing (SA), Efficient Branch and Bound Algorithm (EBBA) and CASP7

PDB	Size	SS &	BCO				SA			EBBA		CASP7
id		energy	$Q(p^*)$	RMSD(p^*)	GDT(p^*) (%)	GDT($p^\dagger$) (%)	$Q(p^*)$	GDT(p^*) (%)	GDT($p^\dagger$) (%)	$Q(p^*)$	RMSD(p^*)	GDT(p^*)
1FC2	43	Pred.	3.65	6.62	52.33	55.23	3.76	47.67	58.14	5.26	8.4	–
		Exact	1.94	1.65	83.72	84.30	2.62	66.28	79.07	4.34	6.6	–
1ENH	54	Pred.	4.67	6.99	40.28	50.93	4.91	40.28	50.46	5.70	10.2	–
		Exact	2.91	2.28	71.30	73.61	3.56	54.63	67.13	4.36	3.5	–
2GB1	56	Pred.	5.41	8.86	30.80	41.96	5.50	29.46	42.41	6.22	7.8	–
		Exact	5.52	9.18	31.70	47.32	5.03	27.68	49.11	4.22	4.3	–
2CRO	65	Pred.	3.85	8.76	31.15	42.31	4.44	35.38	39.62	5.89	9.4	–
		Exact	6.10	7.61	35.38	47.69	6.13	41.54	51.54	6.49	9.2	–
1CTF	68	Pred.	5.43	9.01	36.03	38.97	5.74	33.46	37.87	5.84	11.3	–
		Exact	5.67	7.50	38.60	44.12	5.83	25.74	49.63	7.19	11.0	–
4ICB	76	Pred.	4.77	9.02	32.57	38.49	5.32	29.28	44.08	6.79	6.4	–
		Exact	5.38	10.38	28.29	44.41	5.45	28.95	42.11	6.18	7.4	–
2HG6	106	Pred.	6.14	16.26	14.89	22.17	6.61	17.69	27.59	–	–	30.34%
		Exact	4.70	14.49	20.05	24.29	5.19	19.81	30.19	–	–	–
2J6A	136	Pred.	6.79	14.34	14.34	19.30	6.79	17.10	20.59	–	–	27.78%
		Exact	6.20	16.31	18.38	22.98	7.25	17.46	21.88	–	–	–

At CASP7 the proteins 2HG6 and 2J6A had target numbers T0314 and T0319 respectively. Large values of GDT are preferable whereas low values of RMSD are preferred. Since structure prediction seeks to minimize the energy, $Q(p)$ should be as low as possible. p^* is the structure, encountered during search, with lowest energy and $p^\dagger$ is the one with highest GDT. The same combinatorial protein representation is used for BCO and SA. An identical representation is used for EBBA but some parameters diverge

Table 2 Comparison of best energy values for BCO, SA and EBBA when run on 1FC2, 1ENH, 2GB1, 2CRO, 1CTF and 4ICB

	BCO	SA	EBBA
Average $Q(p^*)$	4.61	4.86	5.71
Improvement over EBBA	24%	17%	–
Improvement over SA	5%	–	–

Note that some parameters diverge in EBBA's representation of the protein and EBBA is the only algorithm that guarantees a globally optimal p^*

reasons why we do not find these very low energy structures for the other proteins. One reason could be that native-like structures cannot be represented accurately enough in our model when trying to represent large proteins. The other possibility is that our search algorithm requires more time to find the native-like structure. This is a subject for further investigation. We did perform a fair amount of ad-hoc experiments adjusting the parameters of the search-methods but no results indicated that any parameters were better suited for large proteins than for small.

References

1. Abbass, H.A.: MBO: marriage in honey bees optimization—a haplometrosis polygynous swarming approach. In: Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, pp. 207–214 (2001)
2. Bahamish, H.A.A., Abdullah, R., Salam, R.A.: Protein conformational search using Bees Algorithm. In: Asia International Conference on Modelling and Simulation, pp. 911–916 (2008)
3. Boberg, J., Salakoski, T., Vihinen, M.: Selection of a representative set of structures from Brookhaven protein data bank. Proteins **14**(2), 265–76 (1992)
4. Chothia, C., Lesk, A.M.: The relation between the divergence of sequence and structure in proteins. EMBO J. **5**, 823–826 (1986)
5. Hamelryck, T.: An amino acid has two sides: a new 2D measure provides a different view of solvent exposure. J. Proteins Struct. Funct. Bioinf. **59**(1), 38–48 (2005)
6. Hamelryck, T., Kent, J.T., Krogh, A.: Sampling realistic protein conformations using local structural bias. PLOS Computat. Biol. **2**, e131 (2006)
7. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. Oper. Res. **37**(6), 865–892 (1989)
8. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes Univ., Engineering Faculty, Computer Engineering Department (2005)
9. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. J. Glob. Optim. **39**(3), 459–471 (2007)
10. Kryshtafovych, A., Fidelis, K., Moult, J.: Progress from CASP6 to CASP7. Proteins Struct. Funct. Bioinf. **69**(S8), 194–207 (2007)
11. Li, Z., Scheraga, H.A.: Monte Carlo-minimization approach to the multiple-minima problem in protein folding. Proc. Natl. Acad. Sci. **84**(19), 6611–6615 (1987)
12. Mayuko, T.S., Daisuke, T., Chieko, C., Hirokazu, T., Hideaki, U.: Protein structure prediction in structure based drug design. Curr. Med. Chem. **11**(5), 551–558 (2004)
13. McGuffin, L.J., Bryson, K., Jones, D.T.: The PSIPRED protein structure prediction server. Bioinformatics **16**(4), 404–405 (2000)
14. Paluszewski, M., Hamelryck, T., Winter, P.: Reconstructing protein structure from solvent exposure using tabu search. In: Algorithms for Molecular Biology (ALMOB) (2006)
15. Paluszewski, M., Winter, P.: EBBA: efficient branch and bound algorithm for protein decoy generation. Technical report. Department of Computer Science, Univ. of Copenhagen, vol. 08(08) (2008)
16. Paluszewski, M., Winter, P.: Protein decoy generation using branch and bound with efficient bounding. In: Proc. of the 8th Int. Workshop, WABI 2008, LNBI 5251, pp. 382–393 (2008)

17. Pham, D., Koc, E., Ghanbarzadeh, A., Otri, S., Rahim, S., Zaidi, M., Phrueksanant, J., Lee, J., Sahran, S., Sholedolu, M., Ridley, M., Mahmuddin, M., Al-Jabbouli, H., Darwish, A.H., Soroka, A., Packianather, M., Castellani, M.: The Bees Algorithm—a novel tool for optimisation problems. In: Proceedings of IPROMS 2006 Conference, pp. 454–461 (2006)
18. Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M.: The Bees Algorithm. Technical report, MEC, Cardiff University, UK (2005)
19. Rohl, C.A., Strauss, C.E., Misura, K.M., Baker, D.: Protein structure prediction using Rosetta. Methods Enzymol. **383**, 66–93 (2004)
20. Sayle, R.: RasMol v2.5 a molecular visualisation program. Biomol. Struc. Glaxo Research and Development Greenford. Roger Sayle and Biomol. Struct. (1994)
21. Simons, K.T., Kooperberg, C., Huang, E., Baker, D.: Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. J. Mol. Biol. **268**(1), 209–25 (1997)
22. Skolnick, J., Kolinski, A., Ortiz, A.R.: MONSSTER: a method for folding globular proteins with a small number of distance restraints. J. Mol. Biol. **265**, 217–241 (1997)
23. Song, J., Takemoto, K., Akutsu, T.: HSEpred: predict half-sphere exposure from protein sequences. Bioinformatics **24**, 1489–1497 (2008)
24. Vilhjalmsson, B., Hamelryck, T.: Predicting a new type of solvent exposure. In: ECCB, Computational Biology Madrid 05, P-C35, Poster (2005)
25. Yuan, Z.: Better prediction of protein contact number using a support vector regression analysis of amino acid sequence. BMC Bioinf. **6**(1), 248 (2005)
26. Zemla, A., Venclovas, C., Moult, J., Fidelis, K.: Processing and analysis of CASP3 protein structure predictions. Proteins **Suppl 3**, 22–29 (1999)
27. Zhang, Y.: I-TASSER server for protein 3D structure prediction. BMC Bioinf. **9**(1), 40 (2008)

5.3 Ranking Beta Sheet Topologies of Proteins

The following 5 pages contains the conference version of our paper "Ranking Beta Sheet Topologies of Proteins. R. Fonseca, G. Helles and P. Winter. Proceedings of WCECS, 2:624-628. 2010" [59].

Ranking Beta Sheet Topologies of Proteins

Rasmus Fonseca*, Glennie Helles* and Pawel Winter*

Abstract—**One of the challenges of protein structure prediction is to identify long-range interactions between amino acids. To reliably predict such interactions, we enumerate, score and rank *all* β-topologies (partitions of β-strands into sheets, orderings of strands within sheets and orientations of paired strands) of a given protein. We show that the β-topology corresponding to the native structure is, with high probability, among the top-ranked. Since full enumeration is very time-consuming, we also suggest a method to deal with proteins with many β-strands.**

The results reported in this paper are highly relevant for *ab initio* protein structure prediction methods based on decoy generation. The top-ranked β-topologies can be used to find initial conformations from which conformational searches can be started. They can also be used to filter decoys by removing those with poorly assembled β-sheets, and finally they can be relevant in contact prediction methods.

Keywords: beta-sheets, protein structure prediction, β-topology

1 Introduction

Predicting the tertiary structure of a protein from its amino acid sequence alone is known as the *protein structure prediction* (PSP) problem. It is one of the most important open problems of theoretical molecular biology. In particular, *ab initio* PSP (especially needed when a homologous sequence cannot be found in the protein data bank) poses a significant problem. One of the reasons why *ab initio* methods struggle is that the conformational space of most protein structure models increases exponentially with the length of the sequence. The complexity of the PSP problem can be reduced using auxiliary predictions such as secondary structures [1, 2, 3, 4], contact maps [5, 3, 6], structural alphabets [7, 8] and local structure predictions [9, 10]. However, all these predictions have a certain level of inaccuracy so they cannot be used to constrain the conformational space, only to guide the conformational search.

A *β-topology* is a partition of β-strands into ordered subsets (each corresponding to a β-sheet) together with the *β-pair* information (pairing of strands and their orientation) for each β-sheet. The order of β-strands within a β-sheet combined with the β-pair information is referred to as the *β-sheet topology*. If the correct β-topology could be predicted, it would, for instance, assist PSP methods to find the native structure [11, 12, 13, 14, 15].

One approach to predict the β-topology of a protein, in the following referred to as the *pair scoring method* [16], is to assign a pseudo-energy to every β-pair. The problem of determining the best β-topology is then formulated as a maximization problem in a complete graph where nodes correspond to β-strands and edge-weights correspond to the pseudo-energy of pairing two strands [12, 16, 17, 18, 15]. Another approach, referred to as the *topology scoring method*, is to enumerate all β-topologies, and to assign a score to each based on the entire β-topology [19, 20]. In general, the β-topology with highest score is assumed to correspond to the correct one [13]. The topology scoring method has also been used to filter decoy sets from Rosetta [19].

Our objective is not to predict the correct topology, but to generate a small set of β-topologies that will, with high probability, contain the correct one. We, therefore, enumerate all β-topologies and use the scoring methods from [16] and [19] to score and rank them. Our experiments show that for a large percentage of examined proteins, the correct β-topology can be found among the 10% top-ranked β-topologies using the pair scoring method (which outperforms the topology scoring method).

Enumerating all β-topologies is a problem for proteins with more than 7 β-strands due to combinatorial explosion. For such proteins, a subset of the β-topologies is enumerated. This subset is guaranteed to contain a β-topology which is consistent with the correct one, meaning that it has no β-pair which does not exist in the correct one. Such β-topologies can be found among the top 10% top-ranked and can also be found for larger proteins.

2 Methods

The following two subsections describe how a set of β-topologies is generated for a given protein, the first for proteins with 7 strands or less and the second for proteins with more strands. The third subsection describes how a score is calculated for each β-topology, and finally the datasets used in the experiments are described.

*{rfonseca, glennie, pawel}@diku.dk. Univ. of Copenhagen, Dept. of Computer Science. Universitetsparken 1, 2100 Copenhagen O, Denmark

ISBN: 978-988-18210-0-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Generating small β-topologies

The secondary structure specifies which amino acids are classified as helix, strand or coil. Continuous segments of strand-classified amino acids are simply referred to as strands.

If the secondary structure has 7 strands or less, all possible β-topologies can be generated by enumerating all valid pairings of strands. A valid pairing of strands is characterized by the following rules: Each strand is paired with one or two other strands and each pair of strands is either parallel or anti-parallel. Table 1 shows how many valid pairings exist in a protein with m strands.

This definition of a valid pairing corresponds largely to the definition of 'overall sheet configuration' used in [21] and 'β-sheet topology' from [16]. It is a representation of the β-topology that does not specify precisely which amino acids form hydrogen bonds in two strands, but it focuses on the overall configuration of the β-pairs in the protein.

m	2	3	4	5	6	7
	2	20	156	1744	23800	373008

Table 1: Number of valid β-topologies.

Generating larger β-topologies

If the secondary structure has 8 strands or more the set of β-topologies is generated the following way. First, a subset of six strands is chosen, and the 23800 corresponding β-topologies are added to the set. This process is repeated for all subsets of 6 strands. A total of

$$\binom{m}{6} \cdot 23800$$

β-topologies are therefore enumerated and scored for proteins with m strands. Table 2 shows this value for $m = 8, \ldots, 13$.

8	9	10	11	12	13
666 400	1 999 200	4 998 000	10 995 600	21 991 200	40 840 800

Table 2: Number of valid β-topologies.

The β-topologies in the final set will contain fewer strands than in the native β-topology. However, it can still be guaranteed that at least one β-topology will be very similar to the native β-topology.

To clarify how a β-topology is compared to the native, we introduce the notions of *native-respecting* and *native-matching* β-topologies. A β-topology is native-respecting if each β-pair corresponds to a β-pair in the native. A β-topology, B, is native-matching if it is native-respecting, and if each β-pair in the native furthermore corresponds to a β-pair in B (i.e., B respects the native and the native respects B). Figure 1 illustrates how β-topologies are compared to the native β-topology. For proteins with more than 7 strands the native-matching topology is never among the generated, but several native-respecting β-topologies will still be among them.

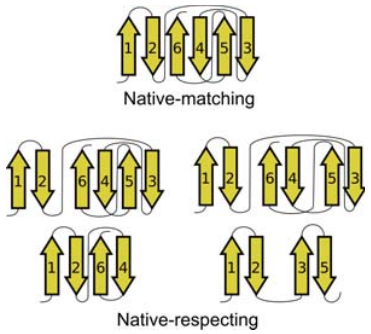


Figure 1: Five β-topologies for 1I8N.

Scoring β-topologies

Two methods of scoring β-topologies have been examined: The topology scoring method and the pair scoring method.

The *topology scoring method* [19], works for proteins with one β-sheet only. It assigns a probability to each β-sheet topology based on the following features: Number of strands, β-pairs, parallel β-pairs, parallel β-pairs with short loops (less than 10 amino acids), jumps (sequential strands that do not form β-pairs), jumps with short loops, the placement of the first strand (near the edge or the center of the sheet) and the helical status of the chain (either all-beta or alpha-beta).

In order to deal with proteins with more than one β-sheet, a more elaborate topology scoring function is needed. In [21], the probabilities of individual β-sheet topologies are combined with two more features, the number of sheets and the number of crossings (consecutive β-strands in different β-sheets), to assign a probability to the entire β-topology.

The *pair scoring method* [16] uses pseudo-energies between pairs of amino-acids in different strands. Neural networks are used to determine these pseudo-energies. The total pseudo-energy of a β-pair is calculated by finding an optimal alignment (either parallel or antiparallel) of the two strands using dynamic programming. The pseudo-energy of the β-pair is then the sum of pseudo-energies for the resulting amino acid pairs. Since a β-topology can be regarded as a set of β-pairs, we calculate the score of a β-topology as the average pseudo-energy of all β-pairs. This ensures that scores of β-topologies are comparable even when they differ in the number of β-pairs.

ISBN: 978-988-18210-0-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

In [22] a third scoring method which is primarily based on hydrophobic packing is discussed. This method, however, is outperformed by both the topology scoring method and the pair scoring method, so the results are not mentioned here.

Datasets

To evaluate how good the scoring of β-topologies is, we generate two datasets. The first is made up of all the chains from PDBSelect25 2009 [23] that contain strands. This is 3305 out of 4423 chains total (75%). Not all the required parameters for the topology scoring method are available in [21], so the dataset is split into a training-set and a test-set (*PDB test-set*), of 161 randomly chosen chains containing between 2 and 7 strands. The training-set is used to learn the parameters in the topology scoring method.

A second test-set, the *CASP8 test-set*, is compiled from all the CASP8 [24, 25] targets that contain strands. This test-set has no guarantee to be as diverse as the PDB test-set, but it gives a good indication of the practical applicability of our method. At CASP8 there were 119 targets of which 13 contained no strands, so the CASP8 test-set consists of 106 protein chains that all have sheets. 53 of the these have between 2 and 7 strands and the majority of the rest contains between 8 and 12 strands.

3 Results and discussion

The primary tool for analyzing sets of β-topologies is a *rank-plot*. The rank-plot for a set of β-topologies shows the rank of each β-topology (x-axis) and its score (y-axis). The set is sorted by non-increasing score. The rank-plot is therefore a monotonically non-increasing curve (see Figure 2). The position of the native-matching β-topology is highlighted with a circle and native-respecting topologies are highlighted with crosses. The average and median rank of native-matching and native-respecting β-topologies will be the primary tool for reporting results. Since there can be more than one native-respecting topology, we only consider the highest ranked.

Ranking small β-topologies

For every protein in the PDB test-set, the secondary structure is extracted from the PDB file and then used to generate a set of β-topologies and the corresponding rank-plot. For 4 out of the 161 proteins in the PDB test-set, a native-matching β-topology was not among the generated β-topologies because one of their strands paired with more than two other strands.

The main question when considering the applicability of enumerating β-topologies is: How many of the top-ranked β-topologies does one have to consider before the native-matching is found? Figure 3 shows how many proteins (percentage) have the native-matching β-topology among the top-ranked. The figure illustrates this for both the topology scoring method (top) and the pair scoring method (bottom). Individual curves are generated for proteins containing the same number of strands. For 80% of all 6 stranded proteins it is sufficient to go through roughly 2230 of the top-ranked β-topologies when using the topology scoring method and 232 when using the pair scoring method. This implies that for a large fraction of proteins, enumerating just a relatively small number (hundreds) of β-topologies, results in a set that has a good chance to contain the native-matching β-topology.

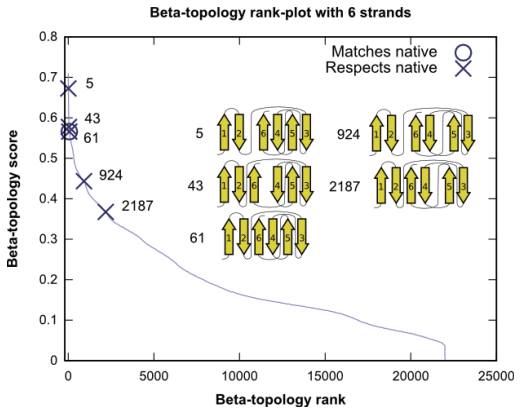


Figure 2: The rank-plot for the all β-topologies of the six-stranded protein 1I8N using the pair scoring method. The native-matching β-topology has rank 61, and the first native-respecting β-topology has rank 5.

The topology scoring method performs well, and at times better, compared to the pair scoring method for proteins with 4 strands or fewer. For proteins with more strands, however, the pair scoring method significantly outperforms the topology scoring method. Therefore, all of the remaining experiments are performed using the pair scoring method.

Table 3 shows more statistics for the rank of the native-matching β-topology using the pair scoring method.

Strands	2	3	4	5	6	7
Proteins	26	33	26	28	27	20
Avg. NM rank	1.08	2.55	4.77	104	213	8850
Median NM rank	1	2	3	49	69	905
Avg. BNR rank	1.08	2.55	1.69	54.3	104	7534
Median BNR rank	1	2	1	13	7	41

Table 3: Average and median ranks of native-matching (NM) and best native-respecting (BNR) β-topologies in PDB test-set.

Ranking larger β-topologies

The native β-topology is among the enumerated for 45% (53 out of 119) of the proteins at CASP8, assuming the secondary structure is predicted correctly. Table 4 (top)

ISBN: 978-988-18210-0-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

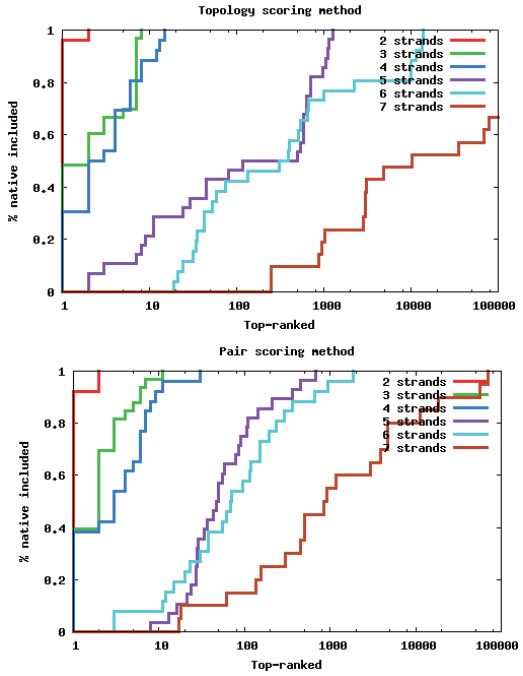


Figure 3: Percentage of native-matching β-topologies among the top-ranked potential topologies using the pair scoring method and the topology scoring method. The x-axis shows the number of top-ranked topologies on a logarithmic scale. The pair scoring method outperforms the topology scoring method for chains with 5 to 7 strands.

shows statistics for the rank of the native-matching and native-respecting β-topologies. Comparing these numbers to those for the PDB test-set in Table 3, it is observed that the ranks of the native-matching β-topologies are higher for the proteins with 6 strands, but notably lower for those with 5 and 7 strands. By comparing the median ranks to the total number of valid β-topologies, shown in Table 1, it is observed that, for a vast majority of the proteins, the native-matching β-topology is among the 10% highest ranked potential β-topologies.

Strands	2	3	4	5	6	7
Proteins	2	4	4	13	11	16
Avg. NM rank	1.5	2.0	5	73	872	4240
Median NM rank	1.5	2.0	2	28	149	768
Avg. BNR rank	1.5	2.0	1.25	31	525	1033
Median BNR rank	1.5	2.0	1.25	4	3	9
Strands	8	9	10	11	12	
Proteins	11	2	7	5	19	
Avg. BNR rank	7628	213	626	6982	11821	
Median BNR rank	59	213	211	464	1582	

Table 4: Average and median ranks of native-matching (NM) and best native-respecting (BNR) β-topologies in CASP8 test-set. For proteins with more than 7 strands, a subset of β-topologies is generated, which is guaranteed to contain a native-respecting β-topology

The ranks of the best native-respecting β-topologies are typically significantly lower than the ranks of the native-respecting. Furthermore, the median ranks are much lower than the average ranks, which indicates that for a majority of proteins the native-respecting β-topology is among the top-ranked, but for a few, the rank is very big.

If, for instance, only the 200 highest ranked β-topologies were considered for each protein in the CASP8 test-set, then the native-respecting β-topology would be among these for 50% of the proteins and the native-matching would be among them for 31%.

4 Conclusions and future work

We presented a method to enumerate β-topologies such that it is guaranteed that a native-respecting β-topology is always among the generated. Furthermore, for proteins with 7 strands or less, a native-matching topology is also guaranteed to be among those generated. The enumerated β-topologies have been scored and ranked using two different scoring methods: The pair scoring method and the topology scoring method. The pair scoring method is shown to outperform the topology scoring method. It is shown that the native-matching β-topology is among the top 10% highest ranked β-topologies, with native-respecting topologies frequently found among the very highest ranked.

There are a number of ways to improve and extend this work. First of all, a better method for scoring β-topologies could be developed by combining the topology scoring method [19] and the pair scoring method [16]. Features and concepts from other sources such as [26, 20, 17, 15] could be used as well. Furthermore, disulphide bindings could be incorporated into the model. This could significantly limit the number of β-topologies for cysteine-containing proteins.

It is assumed that the secondary structure can be predicted correctly. This assumption does not always hold. Particularly the placement of strands is important when enumerating β-topologies. To ensure that at least one β-topology is native-respecting, it should be investigated how the accuracy of strand predictions could be improved.

Finally, the natural extension of this work is to design a PSP method that can use the top-ranked β-topologies to constrain the conformational search and generate high quality protein structure decoys. [14] presents an interesting approach that, using the entire set of β-topologies from [19] and inverse kinematics, can generate high quality decoys. Similar methods, using e.g., only the 200 top-ranked β-topologies, can run longer experiments on each β-topology and possibly give better results for proteins with many strands.

ISBN: 978-988-18210-0-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

References

[1] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, 292:195–202, 1999.

[2] M. Ouali and R. D. King. Cascaded multiple classifiers for secondary structure prediction. *Prot. Sci.*, 9:1162–1176, 2000.

[3] J. Cheng, A. Z. Randall, M. J. Sweredoski, and P. Baldi. Scratch: a protein structure and structural feature prediction server. *Nucl. Acids Res.*, 33:W72–W76, 2005.

[4] C. Cole, J. D. Barber, and G. J. Barton. The Jpred 3 secondary structure prediction server. *Nucl. Acids Res.*, 36:W197–W201, 2008.

[5] R. M. MacCallum. Striped sheets and protein contact prediction. *Bioinformatics*, 20 Suppl 1, 2004.

[6] A. N. Tegge, Z. Wang, J. Eickholt, and J. Cheng. NNcon: Improved protein contact map prediction using 2D-recursive neural networks. *Nucl. Acids Res.*, 37(37):W315–W318, 2009.

[7] C. Etchebest, C. Benros, S. Hazout, and A. G. de Brevern. A structural alphabet for local protein structures: improved prediction methods. *Proteins*, 59:810–827, 2005.

[8] M. Tyagi, A. Bornot, B. Offmann, and A. G. de Brevern. Protein short loop prediction in terms of a structural alphabet. *Comput. Biol. Chem.*, 33:329–333, 2009.

[9] O. Zimmermann and U. H. E. Hansmann. Support vector machines for prediction of dihedral angle regions. *Bioinformatics*, 22:3009–3015, 2006.

[10] G. Helles and R. Fonseca. Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks. *BMC Bioinformatics*, 10:338, 2009.

[11] Y. Cui, R. S. Chen, and W. H. Wong. Protein folding simulation with genetic algorithm and supersecondary structure constraints. *Proteins*, 31:247–257, 1998.

[12] J. L. Klepeis and C. A. Floudas. ASTRO-FOLD: a combinatorial and global optimization framework for Ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. *Biophys. J.*, 85:2119–2146, 2003.

[13] G. Porwal, S. Jain, S. D. Babu, D. Singh, H. Nanavati, and S. Noronha. Protein structure prediction aided by geometrical and probabilistic constraints. *J. Comput. Chem.*, 28:1943–1952, 2007.

[14] N. Max, C. Hu, O. Kreylos, and S. Crivelli. BuildBeta-A system for automatically constructing beta sheets. *Proteins*, 78:559–574, 2009.

[15] R. Rajgaria, Y. Wei, and C. A. Floudas. Contact prediction for beta and alpha-beta proteins using integer linear optimization and its impact on the first principles 3D structure prediction method ASTRO-FOLD. *Proteins*, Early View, 2010.

[16] J. Cheng and P. Baldi. Three-stage prediction of protein beta-sheets by neural networks, alignments and graph algorithms. *Bioinformatics*, 21 Suppl 1:75–84, 2005.

[17] J. Jeong, P. Berman, and T. M. Przytycka. Improving strand pairing prediction through exploring folding cooperativity. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 5:484–91, 2008.

[18] M. Lippi and P. Frasconi. Prediction of protein beta-residue contacts by Markov logic networks with grounding-specific weights. *Bioinformatics*, 25:2326–33, 2009.

[19] I. Ruczinski, C. Kooperberg, R. Bonneau, and D. Baker. Distributions of beta sheets in proteins with application to structure prediction. *Proteins*, 48:85–97, 2002.

[20] A. S. Fokas, I. M. Gelfand, and A. E. Kister. Prediction of the structural motifs of sandwich proteins. *Proc. Nat. Acad. Sci. USA*, 101:16780–16783, 2004.

[21] I. Ruczinski. *Logic regression and statistical issues related to the protein folding problem*. PhD thesis, Univ. of Washington, 2002.

[22] J. L. Klepeis and C. A. Floudas. Prediction of beta-sheet topology and disulfide bridges in polypeptides. *J. Comput. Chem.*, 24:191–208, 2003.

[23] S. Griep and U. Hobohm. PDBselect 1992-2009 and PDBfilter-select. *Nucl. Acids Res.*, 38:D318–319, 2010.

[24] S. Shi, J. Pei, R. I. Sadreyev, L. N. Kinch, I. Majumdar, J. Tong, H. Cheng, B. Kim, and N. V. Grishin. Analysis of CASP8 targets, predictions and assessment methods. *Database*, 2009(0):bap003–, April 2009.

[25] M. L. Tress, I. Ezkurdia, and J. S. Richardson. Target domain definition and classification in CASP8. *Proteins*, 77 Suppl 9(S9):10–17, 2009.

[26] J. A. Siepen, S. E. Radford, and D. R. Westhead. Beta edge strands in protein structure prediction and aggregation. *Protein Sci*, 12(10):2348–59, 2003.

ISBN: 978-988-18210-0-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

5.4 Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction

The following 13 pages contains the published version of our paper "Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction. R. Fonseca, G. Helles and P. Winter. Journal of Mathematical Modelling and Algorithms, 10(4): 357-369. 2011" [60].

J Math Model Algor (2011) 10:357–369
DOI 10.1007/s10852-011-9162-4

Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction

Rasmus Fonseca · Glennie Helles · Pawel Winter

Received: 31 August 2010 / Accepted: 31 August 2011 / Published online: 21 September 2011

Abstract One reason why ab initio protein structure predictors do not perform very well is their inability to reliably identify long-range interactions between amino acids. To achieve reliable long-range interactions, *all* potential pairings of β-strands (β-topologies) of a given protein are enumerated, including the native β-topology. Two very different β-topology scoring methods from the literature are then used to rank all potential β-topologies. This has not previously been attempted for any scoring method. The main result of this paper is a justification that one of the scoring methods, in particular, consistently top-ranks native β-topologies. Since the number of potential β-topologies grows exponentially with the number of β-strands, it is unrealistic to expect that all potential β-topologies can be enumerated for large proteins. The second result of this paper is an enumeration scheme of a *subset* of β-topologies. It is shown that native-consistent β-topologies often are among the top-ranked β-topologies of this subset. The presence of the native or native-consistent β-topologies in the subset of enumerated potential β-topologies relies heavily on the correct identification of β-strands. The third contribution of this paper is a method to deal with the inaccuracies of secondary structure predictors when enumerating potential β-topologies. The results reported in this paper are highly relevant for ab initio protein structure prediction methods based on decoy generation. They indicate that decoy generation can be heavily constrained using top-ranked β-topologies as they are very likely to contain native or native-consistent β-topologies.

Keywords Beta-sheets · Protein structure prediction · Topology

R. Fonseca · G. Helles · P. Winter (✉)
Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
e-mail: pawel@diku.dk

R. Fonseca
e-mail: rfonseca@diku.dk

G. Helles
e-mail: glennie@diku.dk

1 Introduction

Predicting the tertiary structure of a protein from its amino acid sequence alone is known as the *protein structure prediction* (PSP) problem. It is one of the most important open problems of theoretical molecular biology. In particular, ab initio PSP (especially needed when a similar amino acid sequence with known structure cannot be found in the protein database) poses a significant problem. One of the reasons why ab initio methods struggle is that the conformational space of most protein structure models increases exponentially with the length of the primary sequence. The complexity of the PSP problem can be reduced using auxiliary predictions such as secondary structures [2, 3, 9], contact maps [2, 12, 20] or local structure predictions [7, 21]. However, all these predictions have a certain level of inaccuracy so they cannot be used to constrain the conformational space, only to guide the search.

The native *β-topology* of a protein is a partition of β-strands into ordered subsets (each corresponding to a β-sheet) together with the *β-pair* information (indices of paired strands and their orientation in β-sheets)[1] The order of β-strands within a single β-sheet combined with the β-pair information is referred to as the *β-sheet topology*. If the native β-topology could be correctly predicted, it would reduce the search space of PSP and greatly improve the quality of the generated solutions [4, 10, 13, 15, 16]. Furthermore, some PSP methods, such as BuildBeta [13], cleverly use the spatial constraints that a β-topology supplies and can generate a reasonable structure in as little as 10 seconds.

The *pair scoring method* [1] identifies a good β-topology of a protein by assigning a pseudo-energy to every β-pair. The problem of determining the best β-topology is then formulated as a maximization problem in a complete graph where nodes correspond to β-strands and edge-weights correspond to the pseudo-energy of pairing two strands. The problem is to cover all vertices by disjoint paths (corresponding to β-sheets) and cycles (corresponding to β-barrels). Several other variants of this approach have been suggested [8, 10, 11, 16].

The *topology scoring method* [18] enumerates all *β-topologies*, and assigns a score to each based on properties of the entire β-topology. This can be properties such as the number of hairpin turns and parallel β-pairs. In general, the β-topology with highest score is assumed to correspond to the native [15]. The topology scoring method is also used to filter decoy sets from Rosetta [18].

Since the correct β-topology cannot be predicted accurately using either of these methods, we suggest a different approach: All β-topologies are enumerated and the pair scoring method and the topology scoring method are used to score and rank them. Our experiments show that for a large percentage of examined proteins, the native β-topology can be found among the 10% top-ranked β-topologies using the pair scoring method (which outperforms the topology scoring method). An often used step when solving the PSP problem is to generate a set of decoy structures. Using each of the ranked β-topologies as a constraint (one at a time), a set of decoy structures can be constructed. At least one of these decoy structures will be

[1] It is assumed here that a β-strand can have at most two partners. This is often true, but β-strands with more than two partners also exist; such β-strands cannot be dealt with by our method.

generated using the native β-topology, and will therefore have a very high quality. In this study the focus is on the enumeration and ranking of β-topologies, not on generating decoys.

There are three serious problems with the suggested approach. First of all, the correct secondary structure has to be known. One solution to this is to use predicted secondary structures. This leads to the second problem; secondary structure predictors are not always fully reliable. They sometimes over- or under-predict β-strands. In such cases, the native β-topology will not necessarily be among those enumerated. Thirdly, even if the prediction of β-strands is correct, the number of β-strands may be so large that the combinatorial explosion will make it impossible to enumerate all β-topologies. In fact, such combinatorial explosion occurs already when eight β-strands are present.

In order to deal with these problems, the notion of a *strand assignment* is introduced. A strand assignment is a set of non-overlapping intervals that specify which parts of the chain are classified as β-strands. One of the best secondary structure predictors, PSIPRED [9], assigns to each amino acid the probability of it being in a strand (pE-levels), helix and in a coil. Amino acids having pE-levels higher than both helix- and coil probabilities are classified as belonging to β-strands. This results in the *predicted strand assignment*. The main reasons why predicted strand assignments differ from the correct ones are over- and under-predictions of strands. A typical example of under-prediction of strands is shown in Fig. 1.

Since the correct strand assignment cannot always be predicted accurately, we suggest a different approach. Using the pE-levels from PSIPRED, *candidate strands* are suggested and potential strand assignments are enumerated and ranked as described in the Methods section. The problem of combinatorial explosion is dealt with by introducing two limitations when generating the set of potential strand assignments. First, only up to 15 candidate strands are considered in the enumeration procedure. Second, only potential strand assignments with up to seven strands are generated.

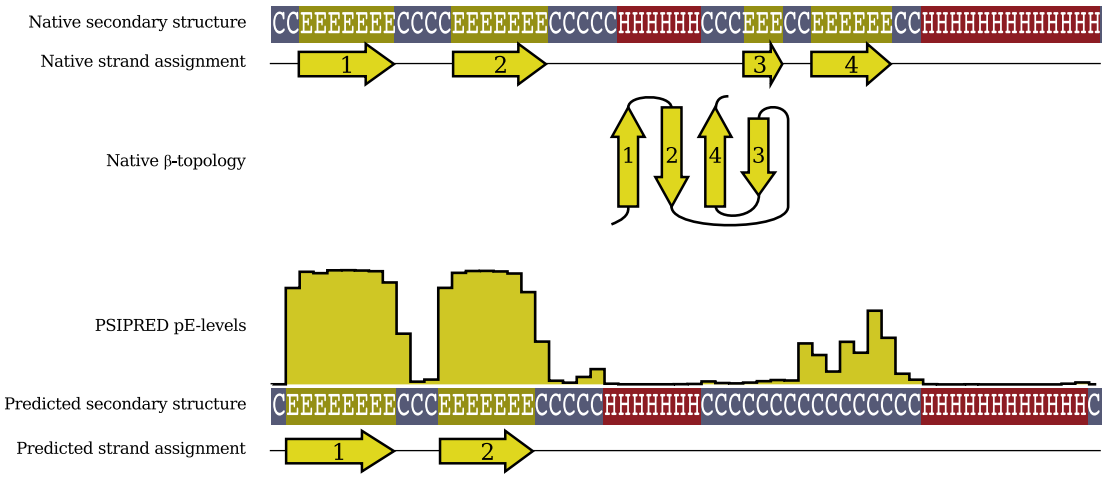


Fig. 1 Comparison of native and predicted strand assignment for the second domain in 3DEV. This example is from CASP8 and is a typical example of β-strand under-prediction. PSIPRED's pE-levels, however, still indicate the presence of a possible strand where the fourth strand should be (although coil probabilities were higher in this region)

Consequently, there will be proteins with eight or more strands whose native β-topologies cannot be generated. However, enumerating potential strand assignments and β-topologies is still relevant for such proteins. To illustrate this, the concepts of *native-respecting strand assignment* and *native-respecting β-topology* are defined. Every β-strand in a native-respecting strand assignment is present in the native strand assignment as well (though the native strand assignment may have more strands). Similarly, every β-pair in a native-respecting β-topology is present in the native β-topology as well (though the native β-topology may contain more β-pairs). For proteins with many strands, a native-respecting strand assignment with up to seven strands can always be found among the potential strand assignments. For most of these, a native-respecting β-topology will be generated. Even though a native-respecting β-topology does not impose as strong a constraint on the PSP problem as a native β-topology itself, it is still a valid constraint that can reduce the search space significantly.

The results reported in this paper are highly relevant for the PSP methods where decoy generation can be constrained or filtered by top-ranked β-topologies. It can also be used in more elaborate contact prediction methods [2, 16, 20].

2 Methods

In the first two subsections the methods for generating potential β-topologies and for calculating their scores are described. Next, it is described how potential strand assignments are generated and how scores are assigned to each of them. The last two subsections describe how to compare both strand assignments and β-topologies and which data sets are used to assess the methods.

2.1 Generating Potential β-topologies

β-strands are numbered $1, 2 \ldots m$ according to the order they appear in the chain. A potential β-topology generated from a strand assignment with m strands is represented using a binary *β-topology-matrix*, $[a_{ij}]_{m \times m}$. Strands i and j form a parallel pair iff $(a_{ij} = 1) \wedge (i > j)$. They form an antiparallel pair iff $(a_{ij} = 1) \wedge (i < j)$. Entries with 1 in the upper (respectively lower) triangle of the matrix therefore represent antiparallel (respectively parallel) pairs. All other entries are 0. A valid β-topology-matrix is characterized by the following three rules: No strand is paired to itself, no pair of strands is paired both parallel and antiparallel and every strand has one or two partners. Given m strands, the complete set of valid β-topology-matrices is generated beginning with the 0-matrix and adding 1's starting at the top row, from left to right (backtracking when necessary).

Table 1 shows the number of valid potential β-topologies, $V(m)$, for up to seven strands. For a single β-sheet with m strands there are $m!/2$ possible orderings of the strands and $2^m/2$ possible combinations of orientations (ignoring symmetric orderings and orientations). This gives a total of $m! \times 2^{m-2}$ possible β-topologies that contain only a single sheet (not counting barrels). Since this number is a lower bound on $V(m)$, it is clear that $V(m)$ grows exponentially with m. For this reason, it is infeasible to enumerate all potential β-topologies for $m \geq 8$.

Table 1 Number of valid β-topology-matrices, $V(m)$, and number of β-topology-matrices that need to be enumerated to include the native, $B(m)$

m	2	3	4	5	6	7
$V(m)$	2	20	156	1,744	23,800	373,008
$B(m)$	2	11	30	700	1,900	70,000

$V(m)$ is determined computationally. $B(m)$ is a result of the experiments shown later in Fig. 5

2.2 Assigning Scores to β-topologies

Two methods for assigning scores to β-topologies have been examined. The *topology scoring method* assigns a probability to each β-sheet topology based on several more or less complicated topological features [17, 18]. This probability is used as the score in the topology scoring method. The topological features include among other things the number of sheets, the number of times a chain crosses from one sheet to another as well as the number of parallel and anti-parallel β-pairs. This method was reimplemented and the parameters obtained from a training-set of proteins specified in the next section.

The *pair scoring method* uses a feed-forward neural network to obtain probabilities of pairing two amino acids. Dynamic programming is then used to optimally align pairs of β-strands in the best possible way (both parallel and antiparallel alignments are included). The score of each alignment is the sum of pairing probabilities between amino acids. The pseudo-energy of pairing two strands is the maximum over all such alignments [1]. A score is assigned to a β-topology by taking the average of pseudo-energies of all its β-pairs. The neural networks were downloaded from the authors homepage.

2.3 Generating Potential Strand Assignments

A strand assignment is defined as a set of m non-overlapping intervals, $\{(s_1, e_1) \ldots (s_m, e_m)\}$, indicating which parts of the chain are β-strands. To ensure that the β-topology of a protein's native structure can be represented, it is important that each β-strand is identified correctly. PSIPRED can be used to predict the placement of strands. It produces three probability levels for each amino acid, $a = 1, 2, \ldots, n$, indicating the probability of a being either helix (pH_a), strand (pE_a) or coil (pL_a). If $pE_a > \max\{pH_a, pL_a\}$ then a is classified as belonging to a strand. This method often fails to predict a strand entirely or predicts a strand where there is none. However, when PSIPRED fails to predict a strand there is often a hilltop (a segment with local minima at both ends) in the pE-levels (see Fig. 1). A set of *candidate strands*, representing possible placements of strands, are therefore generated around hilltops in the pE-plot (see Fig. 2a).

Similar to a strand in a strand assignment, the candidate strands, $i = 1, 2, \ldots, m_c$, are defined by the indices of their first and last amino acids: (s_i, e_i). A potential strand assignment is generated from a subset of candidate strands. All the potential strand assignments are generated by using all possible subsets of candidate strands. Potential strand assignments with 1 or 0 strands are omitted, as valid β-topologies must have at least two strands. To avoid the combinatorial explosion, potential strand assignments with eight strands or more are omitted as well. Figure 2b shows all possible strand assignments that can be generated using the coil and helix probability

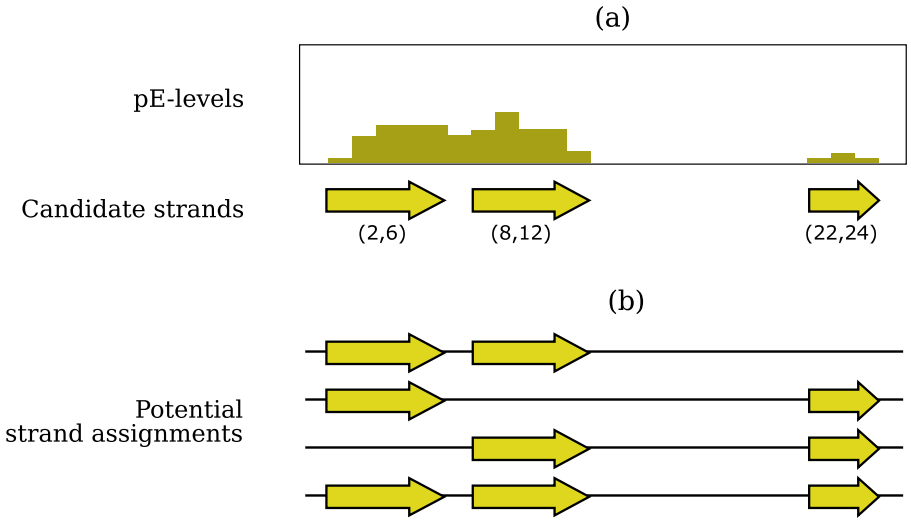


Fig. 2 **a** Probability levels for β-strands (pE-levels of 1ZEC) predicted using PSIPRED. Three candidate strands are identified from the hilltops. **b** All potential strand assignments for 1ZEC

levels and candidate strands from Fig. 2a. The total number of potential strand assignments that are generated for a protein with m_c candidate strands is

$$\sum_{i=2}^{m_c} \binom{m_c}{i} = 2^{m_c} - m_c - 1 \tag{1}$$

2.4 Assigning Scores to Potential Strand Assignments

The pE-levels are used to calculate a score for every potential strand assignment. The average pE value for each strand is calculated as

$$\langle pE \rangle_i = \frac{1}{l_i} \sum_{a=s_i}^{e_i} pE_a \tag{2}$$

where $l_i = (e_i + 1) - s_i$. The score of a strand assignment is then the average of $\langle pE \rangle_i$ for all i, i.e.,

$$\langle pE \rangle = \frac{1}{m} \sum_{i=1}^{m} \langle pE \rangle_i \tag{3}$$

By using averages it is ensured that strand assignments with different number of strands have comparable scores.

2.5 Comparing both Strand Assignments and β-topologies

Two strands, i and j, from different strand assignments are said to overlap iff any part of the interval $[s_i, e_i]$ overlaps $[s_j, e_j]$. Two strand assignments *match* iff there exists a pairing of every strand in the first with every strand in the second such that each pair of strands overlap. A strand assignment is furthermore said to *respect* another strand assignment iff there exists a pairing of every strand in the first with a subset of strands

in the second such that each pair of strands overlaps. This definition will prove useful because potential strand assignments that respect the native strand assignment can be considered 'almost native'. Figures 3 and 4a both give examples of strand assignments that respect another strand assignment.

A β-topology is given by a strand assignment with m strands and a valid β-topology-matrix, $[a_{ij}]_{m\times m}$ specifying which strands are paired. Two β-topologies *match* if their strand assignments match and they have identical β-topologies. Note that if the strand assignments match then the β-topologies will always be of the same dimension. One β-topology, with matrix $[a_{ij}]$, is said to *respect* another, with matrix $[a'_{kl}]$, iff its strand assignment respects that of the second and $(a_{ij} = 1) \Rightarrow (a'_{kl} = 1)$ where i and k are indices of strands that overlap, and j and l are indices of strands that overlap. Figure 3 illustrates how strand assignments and β-topologies are compared. Figure 4b also shows four β-topologies that respect the native.

2.6 Data Sets

For evaluating the quality of the scoring of strand assignments and β-topologies, we generate three data sets. The first two are made up of chains from PDBSelect25 2009 [6] that contain strands. There are 3,305 of these (out of 4,423 chains in total). The topology scoring method is a probabilistic model that has a set of parameters extracted from PDB-files. Not all these parameters are given in [17] so a training-set is needed for the topology scoring method. The proteins from PDBSelect25 2009 are therefore split into a training-set and a test-set (the *PDB test-set*. The *PDB test-set* consists of 161 randomly chosen chains with between two and seven strands and the training-set is the rest of the proteins.

The third data set, the *CASP8 test-set*, is compiled from all the CASP8 [14] targets that contain β-strands. This test-set has no guarantee to be as diverse as PDBSelect25 but gives an indication of the practical applicability of our method. At CASP8 there were 119 targets, but 13 contained no strands, so the CASP8 test-set consists of 106 protein chains that all have β-sheets. 53 of the these have between two and seven strands and the majority of the rest contains between 8 and 12 strands.

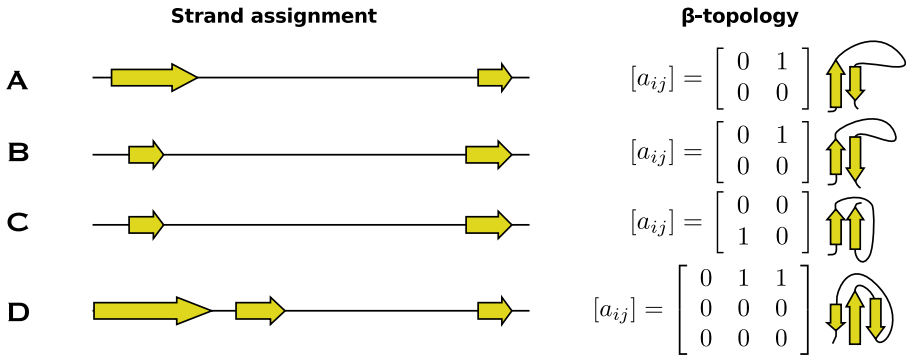


Fig. 3 Examples of comparing strand assignments and β-topologies. Four strand assignments are shown in the *left column*. Strand assignments *A*, *B* and *C* match each other and they all respect *D*. The *right column* shows examples of β-topologies for each strand assignment. The β-topologies *A* and *B* match each other. They neither respect nor match *C* but they both respect *D*

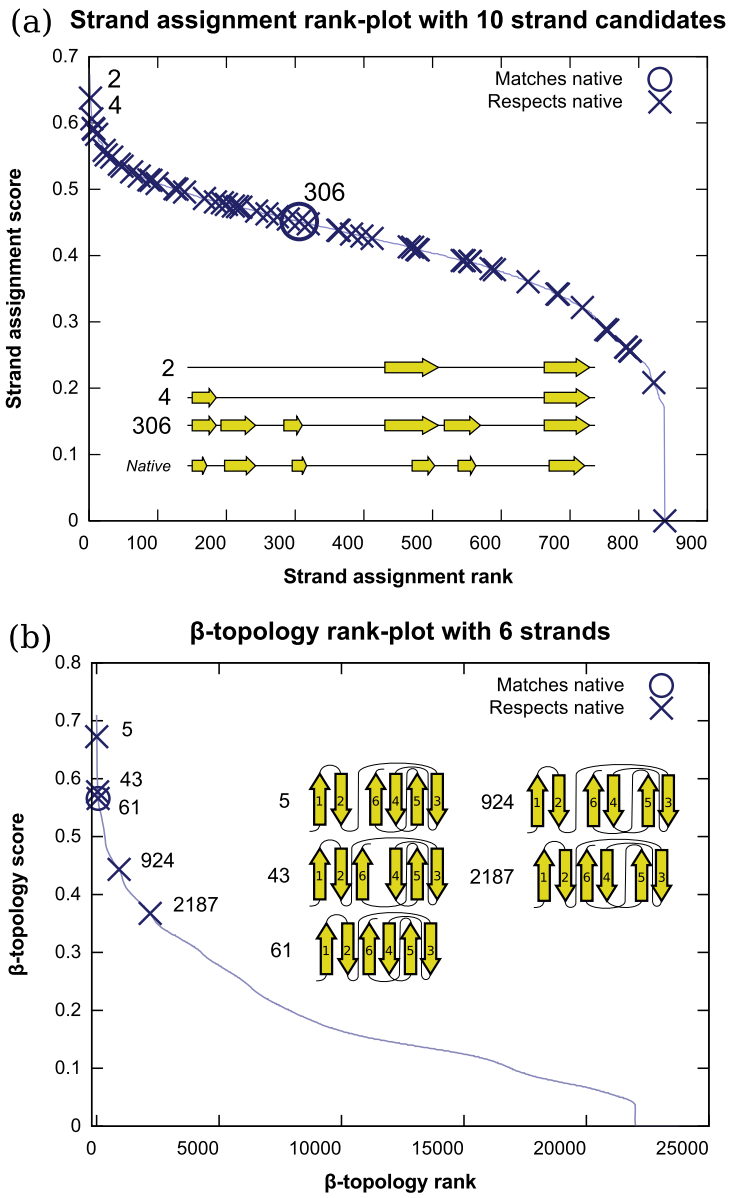


Fig. 4 **a** The strand assignment rank-plot for the six-stranded protein 1I8N. The native strand assignment has rank 306. However, potential strand assignments with ranks as low as 2 and 4 respects the native, and will likely be used to generate β-topologies that respects the native. **b** The β-topology rank-plot for the six-stranded protein 1I8N. The native strand assignment has been used, and the scores are calculated using the pair scoring method. The native β-topology has rank 61, but the β-topology with rank 5 respects the native, and thus provides a constraint that is nearly as good as the native. All topologies that either match or respect the native are highlighted and shown inside the plot

3 Results and Discussion

Given a protein, the *rank-plot* of potential strand assignments illustrates the rank of each strand assignment plotted against its score, as defined in Section 2.4. The rank-plot is therefore a monotonically non-increasing curve as shown in Fig. 4a. The first potential strand assignment that matches the native strand assignment (the *native-matching strand assignment*) is highlighted using a circle. Potential strand assignments that respect the native (*native-respecting strand assignments*) are highlighted using crosses.

Given a protein and a strand assignment, the rank-plot of potential β-topologies illustrates the rank of each β-topology plotted against its score, as defined in Section 2.2 (See Fig. 4b). Only a single β-topology can match the native and only β-topologies with two sheets or more (more than three strands) can respect (and not

match) the native β-topology. These β-topologies are referred to as *native-matching β-topologies* and *native-respecting β-topologies*, respectively.

The average and median rank of native-matching and native-respecting strand assignments and β-topologies will be the primary tool for reporting results.

3.1 Ranking β-topologies Using Native Strand Assignments

An important question when considering the practical applicability of enumerating β-topologies is: How many of the top-ranked β-topologies does one have to enumerate, on average, before the native-matching is found? Using the PDB test-set, Fig. 5 shows how many proteins (percentage) have the native-matching β-topology among the top-ranked. The figure illustrates this for both scoring methods—the topology scoring method and the pair scoring method. Individual curves are generated for proteins containing the same number of strands. For example, for 80% of all 6 stranded proteins it is sufficient to go through roughly 2,230 of the top-ranked β-topologies (out of 23,800 in total) when using the topology scoring method and 232 when using the pair scoring method. This implies that for a large fraction of proteins going through just a relatively small number (hundreds) of β-topologies

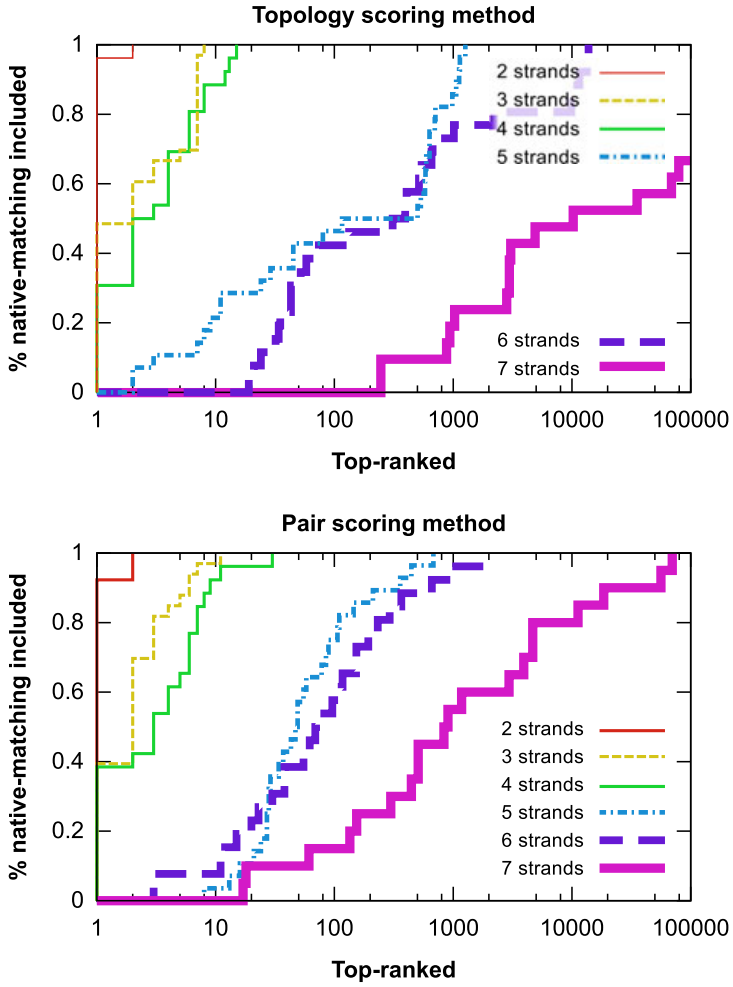


Fig. 5 Percentage of native-matching β-topologies among the top-ranked potential topologies using the topology scoring method and the pair scoring method. The x-axis shows the number of top-ranked topologies on a logarithmic scale

Table 2 Average and median ranks of native-matching β-topologies in PDB test-set (pair scoring method)

Strands	2	3	4	5	6	7
Proteins	26	33	26	28	27	20
Avg. rank	1.08	2.55	4.77	104	213	8,850
Median rank	1	2	3	49	69	905

gives a constraint for the PSP problem that can significantly reduce the size of the conformational search space.

The topology scoring method performs equally well as the pair scoring method for proteins with up to four strands. For proteins with more β-strands, however, the pair scoring method significantly outperforms the topology scoring method. Therefore, all of the remaining experiments are performed using the pair scoring method.

Table 2 shows statistics for the rank of the native-matching β-topology. By comparing the median ranks to the total number of valid β-topologies shown in Table 1 it is observed that, for a vast majority of the proteins, the native β-topology is among the 10% highest ranked potential β-topologies.

3.2 Ranking Potential Strand Assignments

PSIPRED [9] was used to generate pH, pE and pL-levels for all proteins in the PDB test-set. From the pE-levels, candidate strands are identified and potential strand assignments generated. For every potential strand assignment, a score is calculated using the pE-levels, and a rank-plot is generated for every protein (161 in total). The number of potential strand assignments that one has to enumerate before the native-matching strand assignment is encountered is shown in Fig. 6. The red curve converges on ≈81% after 3,000 potential strand assignments (out of approximately 15,000 on average for each protein), which indicates that for only 19% of the proteins in the PDB test-set, no potential strand assignment that matches the native is generated. The typical reason for this is that PSIPRED fails to identify one or more strands. For a majority of the proteins, however, it is enough to enumerate

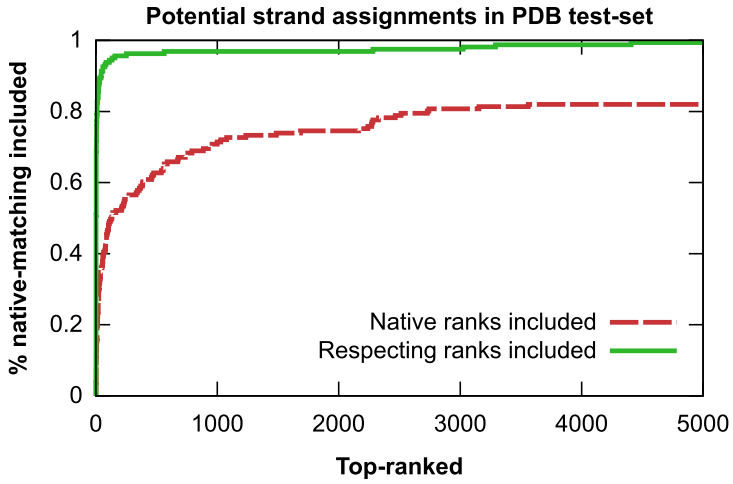


Fig. 6 Percentage of proteins for which the native-matching strand assignment (*red curve*) and native-respecting (*purple curve*) is included among the top-ranked strand assignments

less than 1,000 potential strand assignments. The proteins in the PDB test set have 15,369 potential strand assignments on average. It is therefore observed that for a majority of the proteins, a native-matching strand assignment can be found among the top 7% of the generated strand assignments.

Using only the top-200 ranked potential strand assignments, a native-respecting strand assignment can be found for more than 95% of the proteins.

3.3 Combining Potential Strand Assignments and Potential β-topologies

This subsection seeks to determine the applicability of enumerating both potential strand assignments and potential β-topologies. Since the CASP8 test-set contains proteins that state-of-the-art PSP methods are benchmarked on, we will use this test-set. The combinatorial explosion of β-topologies is dealt with by mainly looking for the native-respecting β-topologies. This ensures that the experiment can be run on proteins with more than seven strands. The experiment seeks to determine how many β-topologies it is necessary to enumerate to find a native-respecting β-topology. It does so without assuming that the native strand assignment is known in advance. Given a potential strand assignment with m strands, $B(m)$ is defined as the number of top-ranked β-topologies which it is necessary to enumerate before the native-matching β-topology is included. The values of $B(m)$ are read off the curves in Fig. 5 and shown in the third row of Table 1. For each of the 106 proteins in the CASP8 test-set, the following experiment is performed: The potential strand assignments are generated, scored and ranked. Starting from the top-ranked potential strand assignment, with m_1 strands, all its β-topologies are generated, scored and ranked. The $B(m_1)$ top-ranked β-topologies are examined. This process is repeated for the lower-ranked strand assignments until the first native-respecting β-topology is encountered. The number of examined β-topologies is then reported. The average and median of these numbers are shown in the second row of Table 3. There is a huge difference between the average number of β-topologies that has to be examined ($\approx$80,000) and the median (44). This indicates that only a limited number of outliers needs to have many β-topologies examined. For a majority of the proteins, less than 50 β-topologies need to be examined before a native-respecting β-topology is found. In many cases, however, this first native-respecting β-topology will only have two strands. This does not provide a very strong constraint on the PSP problem. The experiment above is therefore repeated, but for potential strand assignments

Table 3 Combining potential strand assignments and β-topologies for the CASP8 test-set

Min. m	μ(SA)	$\mu_{\frac{1}{2}}$(SA)	μ(β-sum)	$\mu_{\frac{1}{2}}$(β-sum)
2	102	7	80,634	44
3	271	41	255,956	9,725
4	337	48	361,101	23,586
5	503	198	691,917	242,925

μ(SA) and $\mu_{\frac{1}{2}}$(SA) denotes the average and median rank of the first native-respecting strand assignment from which a native-respecting β-topology can be generated. μ(β-sum) and $\mu_{\frac{1}{2}}$(β-sum) denote the average and median number of β-topologies that have to be examined before a native-respecting β-topology is located. For each row, only topologies with at least 'Min m' strands are considered

with at least three, four and five strands. As a result, approximately 10,000, 22,000 and 240,000 β-topologies, respectively, have to be enumerated for a majority of the proteins before a native-respecting β-topology is found. Although these numbers are high, it is still realistic to generate that many decoys in a reasonable PSP method.

The focus of this subsection has, so far, been solely on native-respecting β-topologies because these can be found for proteins containing any number of strands. The above experiment is repeated for proteins with seven strands or fewer and the number of examined topologies is reported only when the native-matching β-topology is examined. The average and median number of topologies that have to be examined are around 13,900,000 and 4,800,000 and this can only be done for 33 out of the 53 proteins (62%). While these numbers are rather large, any PSP method that efficiently takes advantage of β-topologies, such as [13], will be able to go through that many topologies in a limited amount of time. Furthermore, for a few proteins (3DFD, 3DED, 3DEX, 2KDM and 3DO8), the native β-topology is found after only examining a few thousand β-topologies.

4 Conclusions and Future Work

We have presented a method to enumerate and rank potential β-topologies for proteins with up to seven strands using two different scoring methods: The pair scoring method and the topology scoring method. The pair scoring method is shown to outperform the topology scoring method.

If the correct secondary structure assignment (strand assignment) is not known in advance, the output from PSIPRED is used to generate and rank potential strand assignments with up to seven strands. The results show that the native strand assignment is among the top 7% highest ranked strand assignments for the majority of proteins. Potential strand assignments are then used to generate potential β-topologies. Given the correct strand assignment, it is shown that the native β-topology is among the top 10% highest ranked β-topologies, with native-respecting topologies frequently found among the very highest ranked. Using predicted strand assignments, non-trivial (more than two β-strands) native-respecting β-topologies can be found within the top 10,000 highest ranked β-topologies.

There is a number of ways to improve and extend this work. First of all, a better method for scoring β-topologies could be developed by combining the topology scoring method [18] and the pair scoring method [1]. Features and concepts from other sources such as [5, 8, 16, 19] could be used as well. Furthermore, disulphide bonds could be incorporated into the model. This could significantly limit the number of β-topologies for cysteine-containing proteins.

The results indicate that a relatively large number of strand assignments has to be examined before the native strand assignment is located. The method used for scoring potential strand assignments is very simple. A huge improvement of the results could be achieved by refining the scoring of potential strand assignments using, for instance, machine learning methods like neural networks or support vector machines. Furthermore, a secondary structure predictor that overpredicts strands could also help to ensure that the native strand assignment is among the potential strand assignments for more than 81% of the proteins.

Finally, the very important and natural extension of this work is to design a PSP method that can use the top-ranked β-topologies to constrain the conformational search and generate high quality protein structure decoys.

Acknowledgement We thank Marcus Brazil for his valuable comments and suggestions.

References

1. Cheng, J. Baldi, P.: Three-stage prediction of protein beta-sheets by neural networks, alignments and graph algorithms. Bioinformatics **21**(Suppl 1), 75–84 (2005)
2. Cheng, J., Randall, A.Z., Sweredoski, M.J., Baldi, P.: SCRATCH: a protein structure and structural feature prediction server. Nucleic Acids Res. **33**, W72–W76 (2005)
3. Cole, C., Barber, J.D., Barton, G.J.: The Jpred 3 secondary structure prediction server. Nucleic Acids Res. **36**, W197–W201 (2008)
4. Cui, Y., Chen, R.S., Wong, W.H.: Protein folding simulation with genetic algorithm and super-secondary structure constraints. Proteins **31**, 247–257 (1998)
5. Fokas, A.S., Gelfand, I.M., Kister, A.E.: Prediction of the structural motifs of sandwich proteins. Proc. Natl. Acad. Sci. USA **101**, 16780–16783 (2004)
6. Griep, S. Hobohm, U.: PDBselect 1992–2009 and PDBfilter-select. Nucleic Acids Res. **38**, D318–319 (2010)
7. Helles, G. Fonseca, R.: Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks. BMC Bioinformatics **10**, 338 (2009)
8. Jeong, J., Berman, P., Przytycka, T.M.: Improving strand pairing prediction through exploring folding cooperativity. IEEE/ACM Trans. Comp. Bio. Bioinf. **5**, 484–491 (2008)
9. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices'. J. Mol. Biol. **292**, 195–202 (1999)
10. Klepeis, J.L. Floudas, C.A.: ASTRO-FOLD: a combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. Biophys. J. **85**, 2119–2146 (2003)
11. Lippi, M. Frasconi, P.: Prediction of protein beta-residue contacts by Markov logic networks with grounding-specific weights. Bioinformatics **25**, 2326–2333 (2009)
12. MacCallum, R.M.: Striped sheets and protein contact prediction. Bioinformatics **20**(Suppl 1), i224–i231 (2004)
13. Max, N., Hu, C., Kreylos, O., Crivelli, S.: BuildBeta—a system for automatically constructing beta sheets. Proteins **78**, 559–574 (2009)
14. Moult, J., Fidelis, K., Kryshtafovych, A., Rost, B., Tramontano, A.: Critical assessment of methods of protein structure prediction—Round VIII. Proteins **77**(S9), 1–4 (2009)
15. Porwal, G., Jain, S., Babu, S.D., Singh, D., Nanavati, H., Noronha, S.: Protein structure prediction aided by geometrical and probabilistic constraints. J. Comput. Chem. **28**, 1943–1952 (2007)
16. Rajgaria, R., Wei, Y., Floudas, C.A.: Contact prediction for beta and alpha-beta proteins using integer linear optimization and its impact on the first principles 3D structure prediction method ASTRO-FOLD. Proteins **78**(8), 1825–1846 (2010)
17. Ruczinski, I.: Logic Regression and Statistical Issues Related to the Protein Folding Problem. Ph.D. thesis, Univ. of Washington (2002)
18. Ruczinski, I., Kooperberg, C., Bonneau, R., Baker, D.: Distributions of beta sheets in proteins with application to structure prediction. Proteins **48**, 85–97 (2002)
19. Siepen, J.A., Radford, S.E., Westhead, D.R.: Beta edge strands in protein structure prediction and aggregation. Protein Sci. **12**(10), 2348–2359 (2003)
20. Tegge, A.N., Wang, Z., Eickholt, J., Cheng, J.: NNcon: improved protein contact map prediction using 2D-recursive neural networks. Nucleic Acids Res. **37**(37), W315–W318 (2009)
21. Zimmermann, O., Hansmann, U.H.E.: Support vector machines for prediction of dihedral angle regions. Bioinformatics **22**, 3009–3015 (2006)

5.5 Adjustable Chain Trees for Proteins

The following 17 pages contains the published version of our paper "Adjustable Chain Trees for Proteins. P. Winter and R. Fonseca. Journal of Computational Biology. 19(1): 83-99. 2012" [61].

JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 19, Number 1, 2012

Pp. 83–99
DOI: 10.1089/cmb.2010.0320

Adjustable Chain Trees for Proteins

PAWEL WINTER and RASMUS FONSECA

ABSTRACT

A chain tree is a data structure for changing protein conformations. It enables very fast detection of clashes and free energy potential calculations. A modified version of chain trees that adjust themselves to the changing conformations of folding proteins is introduced. This results in much tighter bounding volume hierarchies and therefore fewer intersection checks. Computational results indicate that the efficiency of the adjustable chain trees is significantly improved compared to the traditional chain trees.

Key words: combinatorial optimization, computational molecular biology, protein folding.

1. INTRODUCTION

A CHAIN TREE IS A DATA STRUCTURE FOR FAST CLASH DETECTION and free energy maintenance of folding protein chains. Suppose that during the simulation of the folding process, an attempt to rotate a backbone or a side chain bond is made. Such a rotation should be undone if it results in a clash (either at some intermediate stage during the rotation or at the end of the rotation). Clashing rotations have to be identified and rejected as quickly as possible. If a rotation is not clashing, the free energy of the new conformation has to be estimated. This energy estimate is crucial for the decision if the rotation should be accepted or rejected (note that in many methods, the increase of the free energy is not always causing a rejection).

The chain tree is a binary tree where leaves are atom groups and each node is associated with a transform matrix and a bounding volume. To maintain the conformation of a protein chain during folding, a brute-force method can be employed. It requires $\Theta(n)$ update time for each conformational change and $\Theta(n^2)$ time for each clash detection. Often, grid methods are used to reduce the clash detection time to $\Theta(n)$. Lotan et al. (2004) showed that, using chain trees, the update time can be reduced to $\mathcal{O}(\log n)$ with a $\Theta(n^{4/3})$ clash-detection time. It was also demonstrated that the average CPU-time required to perform a step in a Monte-Carlo search was much lower when using a chain tree compared to using grids.

We suggest a modification of chain trees based on the assumption that portions of folding proteins (such as α-helices and β-strands) are formed relatively early in the folding process. They remain stable throughout many (if not all) iterations of the simulation. Bonds of such subchains can be locked, and the bounding volumes of these subchains will therefore remain unchanged. As a consequence, the chain tree can be rearranged (so that bounding volumes of locked subchains can be made tighter) and rebalanced (so that updates due to rotations of unlocked bonds can be carried out more efficiently). In addition, we exploit the property of peptide planes (backbone atoms between two consecutive C_α-atoms are always in the same plane) to obtain tight bounding volumes at the lower levels of chain trees. We provide computational results that clearly indicate increased efficiency of chain trees when rearrangements and rebalancing is applied.

Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

Relative improvements (cost speed-up over cost in regular chain trees) for tested protein chains on adjustable chain trees compared to regular chain trees were 16–63%. The average relative improvement was 36%. This substantial improvement was achieved without affecting the worst-case asymptotic time and space complexity. Since chain trees are already shown to be superior to grid methods, we do not compare adjustable chain trees to grid methods.

This article is organized as follows: A brief discussion of proteins with special emphasis on issues related to the chain trees is given in Section 2. Chain trees are described in Section 3. Sections 4–6 discuss how the structure of chain trees can be modified in order to exploit special properties of proteins. Bounding volumes used in the experiments are described in Section 7. Computational results are given in Section 8. Finally, concluding remarks and suggestions for further research are collected in Section 9.

2. PROTEINS

A protein is an organic compound made of n amino acids, $A_1, A_2, \ldots, A_n$, arranged in a linear chain. It folds into a three-dimensional (3D) structure referred to as the *native* conformation. Each amino acid is a molecule containing an amine group NH_2, a carboxylic acid group COOH, and a side chain R that varies for each of the 20 different amino acids. The amine group, the carboxyl group, the side chain, and the hydrogen atom are all covalently bonded to the carbon atom, denoted by C_α. The i-th amino acid A_i, $1 \leq i < n$, is joined with its successor A_{i+1} by a peptide bond between the carboxyl group of A_i and the amine group of A_{i+1} (Fig. 1). The carboxyl group is replaced by the C = O group, and the amine group is replaced by the NH group. Finally, a water molecule is formed during this *polymerization* of consecutive amino acids.

Once linked, individual amino acids are referred to as *residues*, and the chain of consecutive N-, C_α-, and C-atoms (sometimes together with the O-atom attached the C-atom and the H-atom attached to the N-atom) is the *backbone* of the protein.

Bond lengths and angles between two consecutive bonds do not change significantly in residues. They are therefore often considered to be constant and are set to their average values, as shown below in Figure 3.

Backbone bonds between N- and C_α-atoms in the same residue are called N-C_α bonds. The angle of the right-handed rotation around the N-C_α bond is called the *phi* (ϕ) torsion angle.

Backbone bonds between C_α- and C-atoms are called C_α-C bonds. The angle of a right-handed rotation around the C_α-C bond is called the *psi* (ψ) torsion angle.

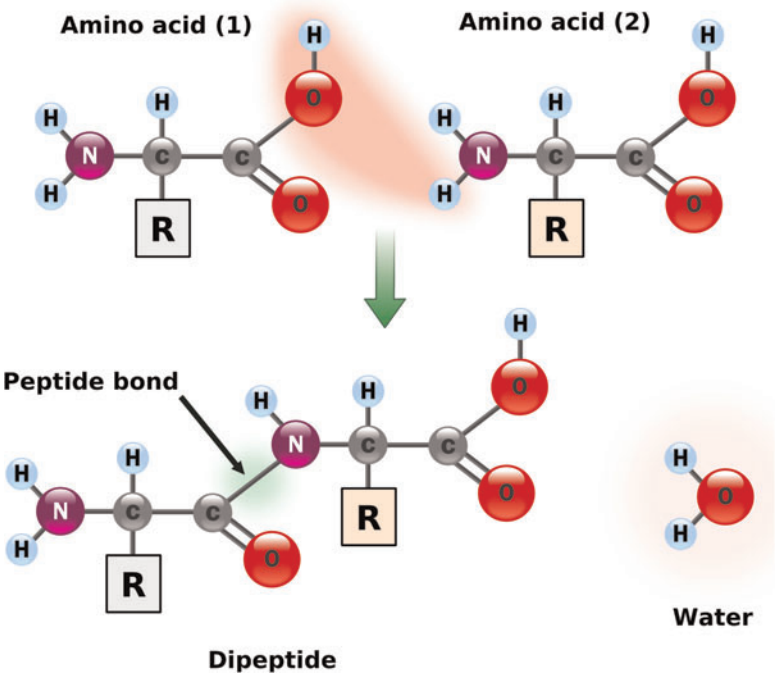


FIG. 1. Peptide formation. Two amino acids form a peptide bond and a water molecule.

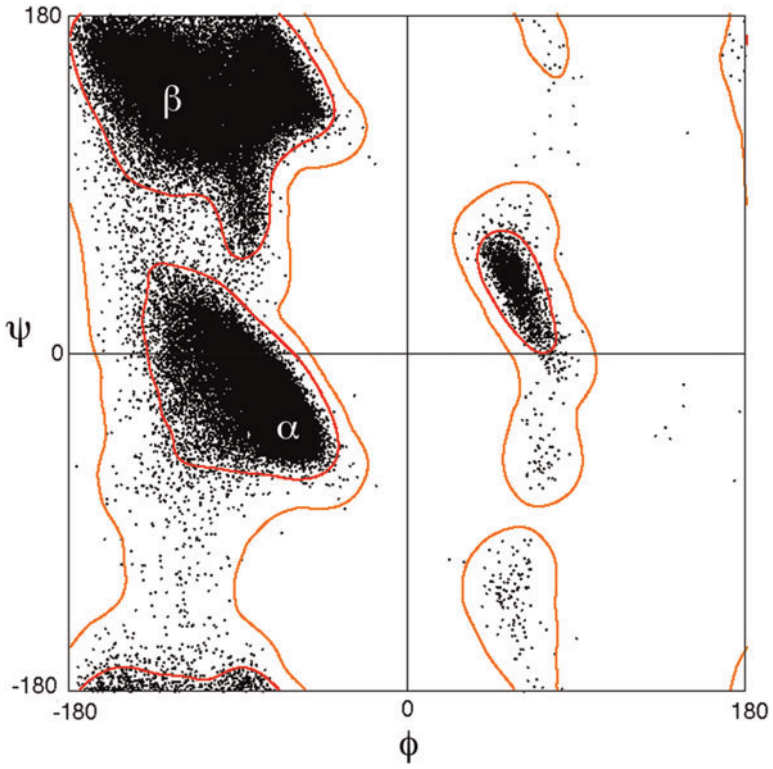


FIG. 2. Ramachandran plot (Wikipedia, 2010). The typical distribution of backbone angles (ϕ, ψ).

Many combinations of ϕ and ψ torsion angles cannot occur, as they would result in steric clashes. The Ramachandran plot (Fig. 2) indicates which combinations of ϕ and ψ occur most frequently.

Bonds between C-atoms of one residue and N-atoms of next residue are called C-N bonds or *peptide* bonds. The angle of right-handed rotation around the C-N bond is called the *omega* (ω) torsion angle. The value of ω is restricted to angles very close to 180° (*trans*-form) but can be close to 0° (*cis*-form) in rare cases (C = O and the NH groups point to the same side). The *trans*-form is about 1000 times more stable than the *cis*-form in all residues but in prolines. The *trans*-form in prolines is only four times more stable than the *cis*-form (Branden and Tooze, 1999). The restrictions of the ω torsion angle imply that backbone atoms between two consecutive C_α-atoms are in the same *peptide plane* (Fig. 3).

3. CHAIN TREES

A protein can be represented by a collection of chain trees; one main chain tree representing the backbone of the protein and *n* small chain trees, each representing a side chain of its residue. The focus of this article is on the efficiency of adjustable chain trees. Consequently, chain trees of side chains have not been implemented, and will only be discussed briefly in the conclusion.

3.1. Backbone chain tree

The *backbone chain tree* is a binary search tree[1] where the leaves correspond to the bonds of the backbone and interior nodes represent subsequences of bonds. Each atom is associated with two bonds. As will be explained in Section 7, bounding volumes will be associated with the leaves and with the interior nodes of the chain tree. As a consequence, bounding volumes of two consecutive nodes will overlap (sharing one atom). This differs from the standard definition of chain trees (Lotan et al., 2004), where leaves represent atoms. Our choice was caused by several factors. First of all, it is intuitively more natural

[1]The term ''search'' is included to indicate the implicit ordering of subchains represented by the nodes: the subchain of the left child of a node directly precedes the subchain of its right child.

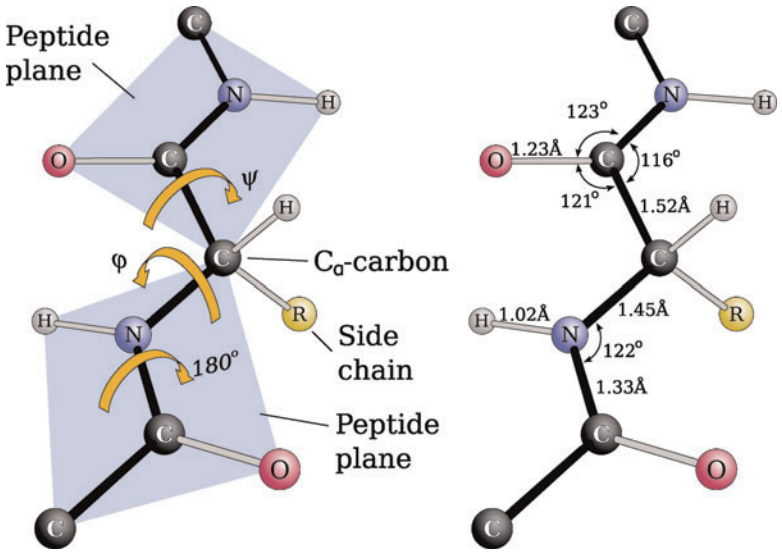


FIG. 3. Backbone geometry and definitions. **(Left)** Since the C-N bond can be fixed at 180°, peptide planes containing six backbone atoms each are formed. **(Right)** Typical backbone bond lengths and angles.

to explain the mechanics of bond rotations when nodes of a chain tree correspond to bonds. Second, we will discuss a modification of chain trees where leaves correspond to entire peptide planes which is a straightforward extension of having bonds as leaves (two consecutive peptide planes share a common C_α-atom). Thirdly, in some applications, such as for example loop closure (Kolodny et al., 2005), where the objective is to get end bonds of subchains to overlap, the representation with leaves representing bonds seems to be more natural. A small overhead caused by the overlap of consecutive bounding volumes is therefore of limited significance. Finally, speed-ups obtained by adjustable chain trees would be of a similar magnitude if their leaves represented atoms rather than bonds.

Any node N of the backbone chain tree is the root of a subtree denoted by $T(N)$. The leaves of $T(N)$ are denoted by $L(N)$. The leaf corresponding to the i-th bond in the backbone is denoted by L_i. The subchain of bonds corresponding to $L(N)$ is denoted by $S(N)$. We say that $T(N)$ *covers* any subset of $S(N)$ and *exactly covers* $S(N)$. When the subtree is obvious from the context, we say that N *covers* $S(N)$. Not every subchain of bonds is exacly covered by a node of a given backbone chain tree.

Given an interior node N in the backbone chain tree, its left and right children are denoted by $l(N)$ and $r(N)$. The parent of N is denoted by $p(N)$.

3.2. *Transformations*

To bring a point p in the coordinate system of the $(i+1)$-th atom of the backbone into the coordinate system of i-th atom of the backbone, the transform matrix R_i is applied to p. This transform matrix R_i is completely defined by a translation vector $\vec{t} = (t_x, t_y, t_z)$ between the two backbone atoms and by the rotation γ of the $(i+1)$-th coordinate system around $\vec{t}$. Let $(x, y, z) = \vec{t}/||\vec{t}||$, $s = \sin(\gamma)$, $c = \cos(\gamma)$, and $d = 1 - c$. Then

$$R_i = \begin{bmatrix} dx^2 + c & dxy - zs & dxz + ys & t_x \\ dxy + zs & dy^2 + c & dyz - xs & t_y \\ dxz - ys & dyz + xs & dz^2 + c & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is the 4×4 transform matrix.

Note that, when performing repeated rotations around the same bond, the number of arithmetic operations can be considerably reduced as the values of x^2, y^2, z^2, xy, xz, and yz remain unchanged. Furthermore, rotations around other bonds do not affect these values.

Consider a point p in the coordinate system of the j-th backbone atom. Its coordinates in the coordinate system of the i-th backbone atom, $i < j$, can be obtained by applying the product R_{ij} of the rotation matrices $R_i, R_{i+1}, \ldots, R_{j-1}$ to p.

The effort of computing the position of the j-th atom in the coordinate system of the i-th atom, $0 \le i < j \le n$, can be reduced from $\mathcal{O}(j - i)$ to $\mathcal{O}(\lceil \log(j - i) \rceil)$ if appropriate transform matrices are

associated with the interior nodes of the chain tree. An interior node N of the backbone chain tree has a 4×4 transform matrix $R_N = R_{l(N)} \times R_{r(N)}$, where $R_{l(N)}$ denotes the transform matrix associated with the left child of N and $R_{r(N)}$ denotes the transform matrix associated with the right child of N. These interior transform matrices can be easily computed by a bottom-up traversal of the backbone chain tree.

Let L_{ij} denote the leaves between L_i and L_{j-1} (both included), and let N be the lowest common ancestor of L_i and L_{j-1}. Note that $L_{ij} \subseteq L(N)$. If $L_{ij} = L(N)$, then $R_{ij} = R_N$. Otherwise, $R_{ij} = R_l \times R_r$ where R_l is determined as follows. If $T(N)$ has L_i as the leftmost leaf, than $R_l = R_{l(N)}$. Otherwise, let R_l initially be a 4×4 identity matrix. Right-multiply R_l by the transform matrices of right children of left-entered nodes when going from L_i to N. R_r is determined in an analogous way.

When a rotation around the bond represented by a leaf L_i has been carried out, only transform matrices between L_i and the root of the backbone chain tree need to be updated. They are determined bottom-up by multiplying transform matrices of their two children.

3.3. Clash detection

Suppose that the bond L_i, $1 < i < n$, of the backbone is rotated by the torsion angle α. To identify a clash, traverse the chain tree from L_i to the root. Let $\mathcal{T}_l$ and $\mathcal{T}_r$ denote two sets of subtrees of the chain tree. Initially, $\mathcal{T}_l = \mathcal{T}_r = \emptyset$. When arriving at an interior node N from its left child $l(N)$, a search for a clash between $S(r(N))$ and any $S(M)$, $M \in \mathcal{T}_l$, is carried out. If no clash is detected, $S(r(N))$ is added to $\mathcal{T}_r$. When arriving at N from its right child $r(N)$, a search for a clash between $S(l(N))$ and any $S(M)$, $M \in \mathcal{T}_r$, is carried out. If no clash is detected, $S(l(N))$ is added to $\mathcal{T}_l$. Note that L_i is neither in $\mathcal{T}_l$ nor in $\mathcal{T}_r$. But the two atoms connected by L_i will be covered by the first subtree added to $\mathcal{T}_l$ and by the first subtree added to $\mathcal{T}_r$.

Rather than traversing the chain tree from L_i to the root, a top-down traversal is a possibility. This would result in $\mathcal{T}_l$ and $\mathcal{T}_r$ initially containing subchains far apart in the backbone. A hybrid method where the path between L_i and the root is traversed from both ends is also a possibility.

Consider two nodes $M \in \mathcal{T}_l$ and $N \in \mathcal{T}_r$ of the backbone chain tree. Note that $L(M) \cap L(N) = \emptyset$. Let $B(M)$ and $B(N)$ denote bounding volumes completely containing M and N, respectively. Different types of boundary volumes have been suggested in the literature. Oriented bounding boxes (Ericson, 2005) are probably most well-known and will also be briefly discussed in Section 7. Their purpose is to simplify clash detection; when two bonding volumes are disjoint, so are the structures bounded by them. Tight bounding volumes with straightforward intersection checks can result in substantial speed ups of clash detection of bounded structures (Fig. 4).

If $B(M) \cap B(N) = \emptyset$, then there is no clash between $S(M)$ and $S(N)$. If $B(M) \cap B(N) \neq \emptyset$, the clash may exist and the search for it has to continue recursively down the backbone chain tree. One possibility is first to search for a clash between $S(l(M))$ and $S(N)$. If no clash is established between these two subchains, a search for a clash continues between $S(r(M))$ and $S(N)$. Instead of splitting $S(M)$, one could of course split

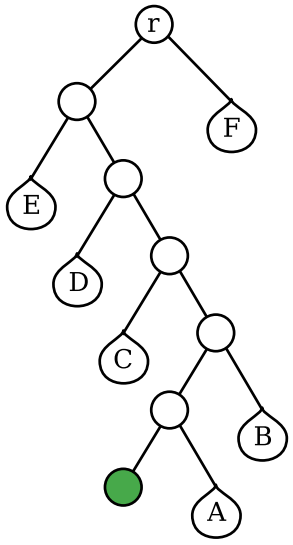


FIG. 4. Clash detection after bond rotation (shaded leaf node). Nodes $A, B, \ldots, F$ denote subtrees, while r is the root of the chain tree. If no clash is detected, then $B_l = \{C, D, E\}$ and $B_r = \{A, B, F\}$ at the end.

$S(N)$. The choice of which of the nodes to split first is governed by the volumes of $B(M)$ and $B(N)$ such that the larger is split first.

4. GROUPING PEPTIDE PLANES

As already mentioned in Section 2, three consecutive bonds—C_α^i - C^i, C^i - N^{i+1}, and N^{i+1} - C_α^{i+1}—are in the same peptide plane because the ω torsion angle on the C^i-N^{i+1} bond is fixed at 180° (or sometimes at 0°) throughout the entire folding process (or when predicting the protein structure). It is therefore advantageous to modify the backbone chain tree so that it will contain nodes exactly covering the three bonds of each peptide plane. Bounding volumes of peptide planes are identical and can be computed by slower but exact methods in the preprocessing phase. Furthermore, some bounding volumes are indeed very tight when applied to peptide planes. Grouping of peptide planes was not previously utilized in the chain trees for protein folding or structure prediction.

When creating a backbone chain tree, three bonds of each peptide plane will be exactly covered by a *peptide subtree* of height two. The root of such a peptide subtree is called a *peptide node*. The left child of the peptide node has two leaves, while the right child is a leaf. Forcing peptide subtrees to be in the backbone chain tree may cause it to become slightly more unbalanced than it would be the case if almost all nodes of height 2 covered four instead of three leaves.

5. LOCKING AND GROUPING SECONDARY STRUCTURES

Suppose that at some stage of the folding process, a portion of the backbone is in its native conformation and none of its bonds will subsequently be rotated. This can, for example, happen when an α-helix or a β-strand is formed. Furthermore, in protein structure prediction, α-helices and β-strands are very often predicted in a preprocessing phase. As a consequence, ideal three-dimensional conformations of such secondary structures can be precomputed and will remain the same in all predicted structures. Their bonds will not be rotated and are said to be *locked*. A consecutive sequence of locked bonds (corresponding to for example an α-helix or a β-sheet) is called a *locked subchain*, and the subtree covering a locked subchain is said to be *locked*.

There are several advantages to rearranging chain trees so that their locked subchains are covered exactly by an internal node. Since all leaves of such a node are locked, tighter bounding volumes of *all* nodes in the subtree can be determined by more elaborate methods. Furthermore the depth of non-locked nodes decreases makes clash detections androtations faster. This rearrangement process is referred to as *grouping*.

Consider for example the top-left chain tree in Figure 6a. Green (shaded) nodes are locked. Their lowest covering node is A (the root of the chain tree), but $T(A)$ does not cover the locked nodes exactly. In the chain tree in Figure 6d, the node A covers locked nodes. A straightforward two-pass iterative grouping algorithm requiring $\mathcal{O}(h)$ time (where h is the height of the regrouped tree) and $\mathcal{O}(1)$ space is described in the remainder of this section.

Consider a chain tree T and assume that it is the smallest subtree containing a locked sequence of leaves $L_{ij} = \{L_i, L_{i+1}, \ldots, L_{j-1}\}$, $1 \le i < j < n$. T has to be grouped so that one node covers L_{ij} exactly. Let $S_l(N)$ (respectively $S_r(N)$) denote the left (respectively, right) swap[2] of the branch between node N and its parent.

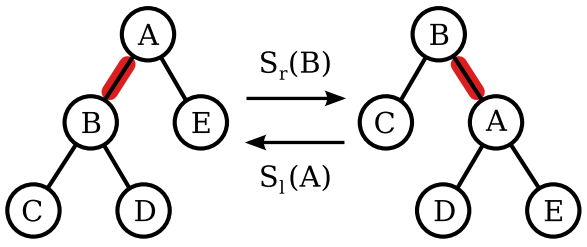


FIG. 5. Left and right swaps.

[2]Swaps are in the binary search tree literature called "rotations", but we wish to avoid confusion with bond rotations.

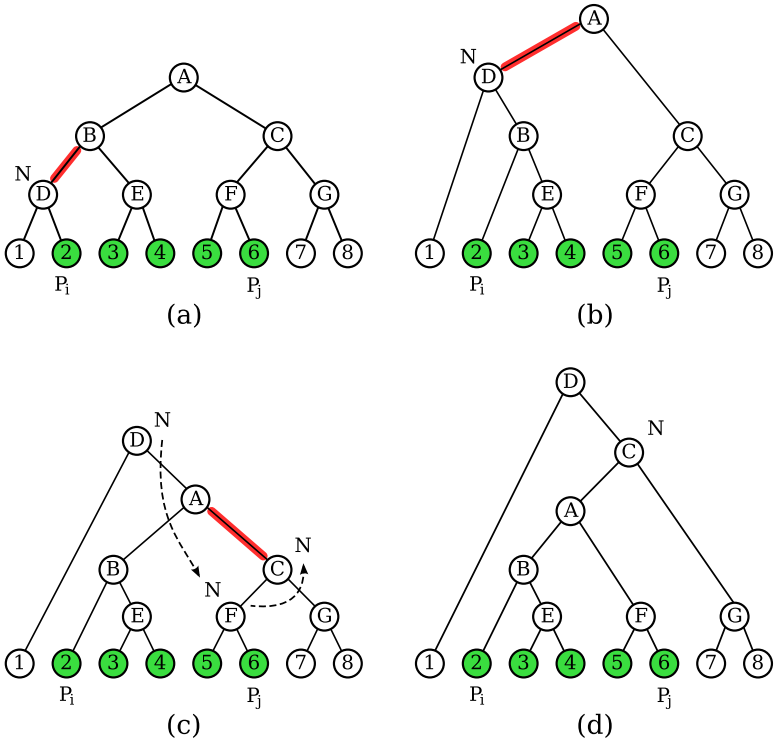


FIG. 6. Example of grouping. **(a)** Initially $N = D$. $6 \notin T(2)$, $2 \in T(2)$ and $D = l(B)$. $S_r(D)$ is applied. **(b)** $N = D$, $6 \notin T(B)$, $2 \in T(B)$ and $D = l(A)$. $S_\tau(D)$ is applied. **(c)** $N = D$. $6 \in T(A)$, and 2 is the leftmost leaf of $T(A)$. First phase completed. $N = F$. $2 \notin T(5)$ and $6 \notin T(5)$. Hence $N = C$. $2 \notin T(F)$, $6 \in T(F)$ and $C = r(A)$. $S_l(C)$ is applied. **(d)** $N = C$ and $T(A)$ has 2 as the leftmost leaf and 6 as the rightmost leaf.

This is permissible only if N is a right (respectively, left) child (Fig. 5). The reader is referred to Cormen et al. (2009) for more on binary search trees and swaps.

Consider the following iterative procedure beginning at the leaf L_i:

$N = p(L_i)$
while $L_{j-1} \notin T(r(N))$ do
  if $L_i \in T(r(N))$
    if $N = r(p(N))$ then $S_l(N)$ else $S_r(N)$
  else $N = p(N)$.

At each iteration, the depth of the node N either reduces by 1 or it remains unchanged. In the latter case, the next iteration reduces the depth of N by 1. As a consequence, N will after a finite number of iterations cover L_{ij}. It is also obvious that L_i remains the leftmost leaf of $T(N)$.

If $T(N)$ exactly covers L_{ij}, the sought grouping is obtained. Assume therefore that L_{j-1} is not the rightmost leaf of $T(N)$.

Let N now denote the parent of L_{j-1}. The iterative procedure needed to make L_{ij} exactly covered is very similar to the one that resulted in N to become the root of the subtree having L_i as a leftmost leaf and being the lowest common ancestor of both L_i and L_{j-1}. The details are therefore omitted.

The discussion in this section can be summarized as follows:

Corollary 1. *A chain tree with k locked leaves can be regrouped in* $\mathcal{O}(1)$ *space and* $\mathcal{O}(h)$ *time where h is the height of the chain tree. If the chain tree is balanced, then* $h \in \mathcal{O}(\log n)$ *where n is the number of leaves.*

6. REBALANCING

When locking and grouping chain trees, they will inevitably become unbalanced in the sense that the heights of the siblings may differ by more than one. There are in fact two types of unbalance that can occur. The less important unbalance may occur in exactly covering trees that emerge after regrouping. This is not so important because bonds of leaves covered by such subtrees will normally not be rotated. But this unbalance after regrouping can still have undesirable influence on clash detection.

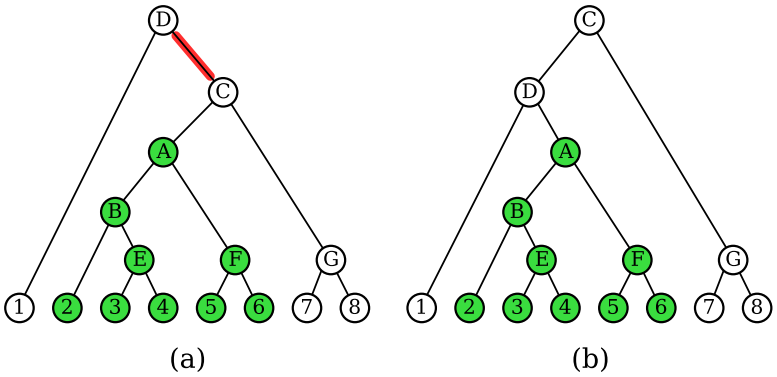


FIG. 7. Rebalancing the chain tree. **(a)** Left child of the root has height 0 while right child has height 2 (subtree rooted at A is locked and therefore regarded as a leaf). **(b)** Left swap on the branch from C to D results in a tree where both children of the root have height 1.

The second type of unbalance occurs when regrouped, exactly covering subtrees are considered as leaf nodes (represented by their roots) Since bounding volumes associated with nodes of such subtrees and, in particular, with their roots can be expected to remain unchanged in subsequent iterations, they can be computed by more precise, slower methods. Search for clashes through such subtrees will therefore occur less often because of tighter bounding volumes. Such subtrees can therefore be pushed further away from the root of the chain tree. Futhermore, this will bring some unlocked leaves closer to the root making the rotations and clash detections less expensive. An example of rebalancing can be seen in Figure 7.

Suppose that a chain tree contains an internal node A of height k, $k \geq 3$. Let $B = l(A)$, $C = r(A)$, $D = l(B)$, and $E = r(B)$. Suppose that $h(B) = k-1$ and $h(C) \leq k-3$. The case $h(B) \leq k-3$ and $h(C) = k-1$ is dealt with in analogous manner (right swaps being replaced by left swaps). Assume furthermore that no other pair of siblings in $T(A)$ has heights differing by 2 or more.

Suppose first that $h(D) = k-2$ and $k-3 \leq h(E) \leq k-2$ (Fig. 8a). Perform the right swap on the branch between B and A. As a consequence, B becomes the root of the subtree, A becomes the right child of B, D becomes the left child of B, while E becomes the left child of A as shown in Figure 8a. If $h(E) - h(C) > 1$, rebalance A again. Notice however that the height of A is at least one less than before the swap. Hence, rebalancing of A may propagate down the chain tree but cannot be repeated more than $\mathcal{O}(k)$ times, and k is bounded by the height of the chain tree.

Suppose next that $h(D) = k-3$ and $h(E) = k-2$ (Fig. 8b). Let $F = l(E)$ and $G = r(E)$. Perform the left swap on the branch from E to B (making E the left child of A) followed by the right swap on the branch from E to A. E becomes the root of the subtree. A becomes the right child of E with G being its left child and C being its right child, as shown in Figure 8b. If $h(G) - h(C) > 1$, rebalance A again. Also in this case, the rebalancing of A can propagate down the chain but cannot be repeated ore than $\mathcal{O}(k)$ times.

The above procedure explains how to perform a rebalancing step given the node A. The rebalancing of the entire chain tree is performed by applying this step to every single node in the chain tree in a bottom-up fashion as a preprocessing step. If, for example, α-helices and β-strands have been formed, their bonds are

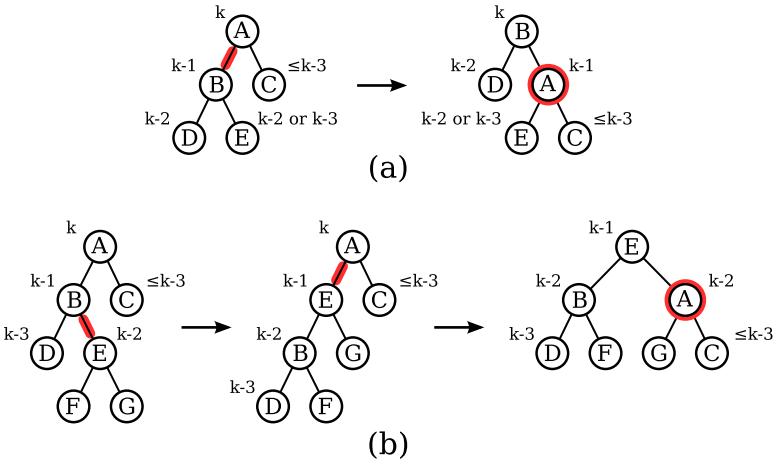


FIG. 8. The two rebalancing cases considered in the rebalancing method. The method ensures that unbalanced nodes are either balanced or propagated downwards in the tree.

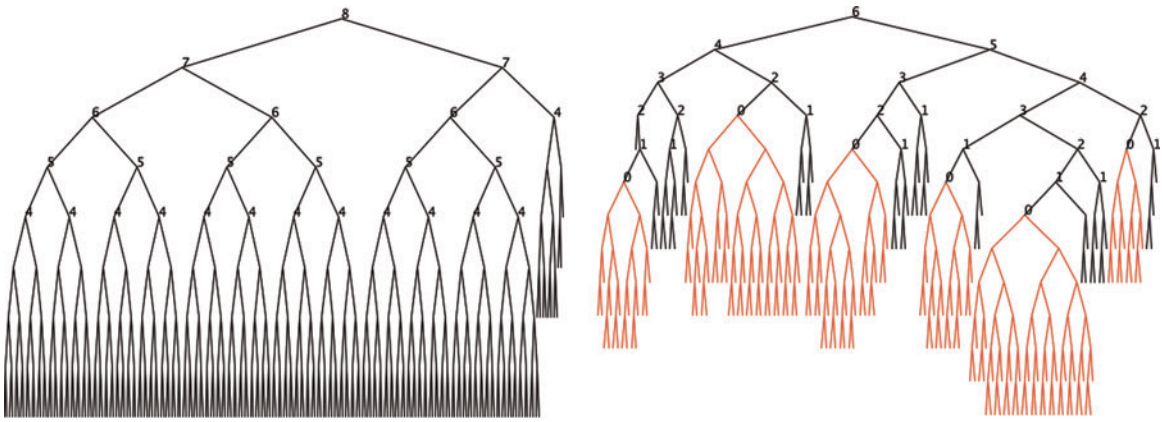


FIG. 9. The chain tree of 1CTFA before and after locking and rebalancing. Red/shaded subtrees are locked subtrees of secondary structures. Integers at nodes are their heights. Note that during the rebalancing the roots of red/shaded subtrees and roots of peptide planes are regarded as leaves of height 0.

locked and the chain tree is regrouped so that all secondary structures are covered exactly. Next, subtrees exactly covering secondary structures are replaced by their roots. The chain tree is rebalanced (identifying siblings with height difference of more than 1 in a bottom-up fashion). Finally, the subtrees exactly covering secondary structures are added back to the chain tree. The chain tree of the protein 1CTFA (see Fig. 12 below) before and after locking, grouping, and rebalancing is shown in Figure 9. The chain tree was also rearranged so that peptide planes are exactly covered in the bottom chain tree (subtrees of height 2 have 3 instead of four leaves).

The discussion in this section can be summarized as follows:

Corollary 2. *Rebalancing of a single node in the chain tree can be done in* $\mathcal{O}(1)$ *space and* $\mathcal{O}(h)$ *time where h is the height of the node. If the chain tree is balanced, then* $h \in \mathcal{O}(\log n)$ *where n is the number of leaves. Rebalancing of all nodes in the chain tree (bottom-up) can then be carried out in* $\mathcal{O}(n \log n)$ *time.*

7. BOUNDING VOLUMES

Use of bounding volumes in chain trees requires efficient methods to:

- transform a bounding volume to another coordinate system,
- decide if two bounding volumes intersect,
- find a tight volume bounding two smaller volumes,
- find a tight volume bounding a set of bonds or atoms.

In this study we decided to use line-segment swept spheres as bounding volumes. A *line-segment swept sphere* (LSS) is the Minkowski sum of an arbitrarily oriented line-segment and a sphere. A LSS is therefore fully specified by the two end-points of the line-segment and the radius of the sphere. A LSS is also sometimes referred to as a capsule, a capped cylinder, a spherocylinder (Ericson, 2005), or even as a cigar (Bereg, 2004).

Our choice was motivated by our intuition that LSSs are much better suited to bound proteins (especially if protein chains are appropriately divided into subchains) than oriented bounding boxes (Gottschalk and Manocha, 1996) and rectangular swept spheres (Larsen et al., 2000; Eberly, 2000) that are more commonly used. Our intuition proved in fact to be correct. The comparative study (Fonseca and Winter, 2010) of various bounding volumes clearly indicated that LSSs are superior (at least for protein chains). The relative speed-up is in fact greater when using either adjustable chain trees with oriented bounding boxes or rectangular swept spheres. In terms of dramatic speed-ups, LSSs are the least spectacular choice. But we nevertheless opted for LSSs, as they are most suitable for protein chains.

A LSS is transformed to another coordinate system by applying an appropriate transform matrix to the end-points of its defining line-segment.

The intersection test between two LSSs is performed as described in Ericson (2005). It basically reduces to deciding if the distance between the two defining line-segments is less than the sum of the radii.

To find a good (but not necessarily optimal) LSS bounding a set of points, the principal component, $\vec{d}$, of the points is used as the direction of the defining line-segment. The points are projected onto a plane normal to $\vec{d}$ and the smallest enclosing circle C with radius r is found. C together with $\vec{d}$, defines an infinitely extended cylinder that contains all the points (now back in 3D). Finally, the infinite line-segment is capped with two hemispheres of radius r such that the points are bounded. Centers of these hemispheres define end-points of the defining line-segment.

To find a good (but not necessarily optimal) LSS, B bounding two smaller LSSs, B_l and B_r, the four spheres defined by the end-points of the line-segments of B_l and B_r and their radii are considered. The direction $\vec{d}$ of B is determined as the direction between the two most remote points on the surfaces of these four spheres. The four spheres are projected onto a plane orthogonal to $\vec{d}$ as four circles (one will be nested in another and can be disregarded). The smallest circle enclosing the remaining three circles together with $\vec{d}$, defines an infinitely extended cylinder containing all four spheres. Finally, the cylinder is capped as explained above.

8. COMPUTATIONAL RESULTS

The main purpose of adjustable chain trees is to deal with α-helices and β-strands (such secondary structures will normally be fixed beforehand or created early in the folding process). In order to justify the need for adjustable chain trees, it is therefore necessary to establish if occurrences of secondary structures are sufficiently common. We argue in Subsection 8.1 that this is indeed the case, and we choose a set of chains for computational experiments. In Subsection 8.2, we set up an appropriate cost measure to evaluate the speed-up independently of implementational and hardware details. In Subsection 8.3, we show to what extent adjustable chain trees speed-up clash detections.

8.1. Data sets

Statistics presented in this section are based on proteins (or rather chains) taken from PDBSelect 25 (Berman et al., 2000). The version we consider contains 4018 chains. Forty-two of these were filtered out because of various format problems.

Lengths of chains in PDBSelect 25 are shown in Figure 10. Frequencies of chains with n residues, $10\times l \le n \le 10\times l + 9$, $l = 0, 1, \ldots, 29$, are averaged. Chains with more than 300 residues are rare, and therefore not shown. It can be seen that chains with 110–119 residues are among the most common.

Figure 11 shows the distribution of chains in PDBSelect 25 with respect to the portion of residues in secondary structures. Out of the 3976 considered chains, only 60 had no secondary structures. More than 75% of all chains had over 50% of their residues in either helices or β-strands.

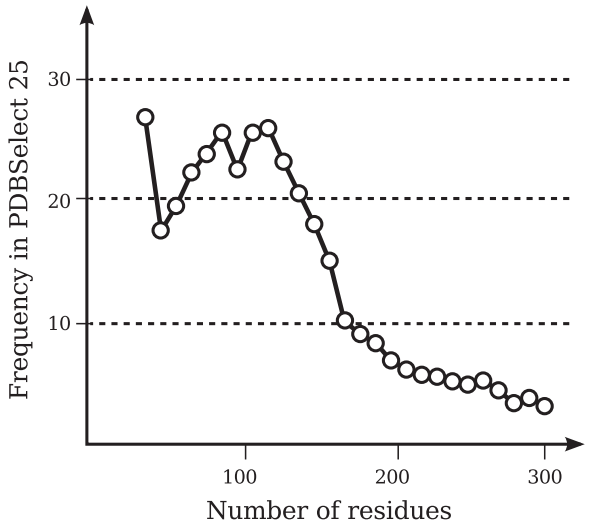


FIG. 10. Distribution of lengths of chains in PDBSelect 25.

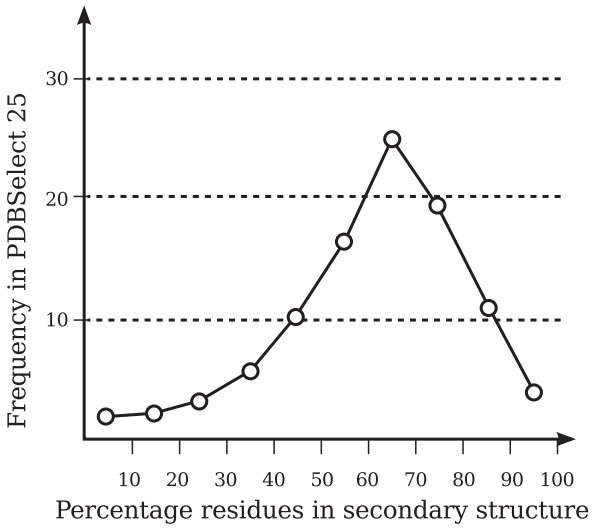


FIG. 11. Percentage of secondary structures in chains of PDBSelect 25.

Based on the above statistics, chain tree experiments were performed on two data sets. The first data set had five chains with 110–119 residues, the typical length in PDBSelect 25. These five chains were selected so that the portions of residues in secondary structures were one in each of the five intervals $[i\times 10\%, (i+1)\times 10\%[$, $i = 4, 5, \ldots, 8$. The first five rows of Table 1 provide some basic characteristics of the five chosen chains.

The second data set is the four chains used in the seminal article on chain trees (Lotan et al., 2004). The lengths vary from 68 to 755 residues. The last four rows of Table 1 indicate which chains are in the second data set. Figure 12 shows all nine examined chains.

8.2. Cost measure

To evaluate how much grouping of peptide planes, locking, regrouping, and balancing improves the efficiency of chain trees, we determined the *average cost* of a single rotation in the chain tree. The cost of a rotation was defined as

$$N_V C_V + N_U C_U + N_P C_P$$

where

- N_V is the number of bounding volume pairs tested for overlap,
- C_V is the cost of testing two bounding volumes for overlap,
- N_U is the number of bounding volumes updated due to rotations,
- C_U is the cost of updating a bounding volume,

TABLE 1. SECOND COLUMN IS THE NUMBER OF RESIDUES IN THE EXAMINED CHAINS

PDB-id	*No. of res.*	*% in αβ*	*#α + #β*
1X5RA	112	42%	2 + 6
1X0OA	119	58%	3 + 5
1XDXA	114	63%	2 + 4
1AKPA	114	79%	2 + 11
1Y6DA	114	82%	7 + 0
1CTFA	68	82%	3 + 3
1LE2A	144	83%	5 + 0
1HTBA	374	52%	10 + 19
1JB0A	755	62%	12 + 4

The third column indicates the portion of residues in secondary structures. The fourth column indicates the number of α-helices and β-strands.

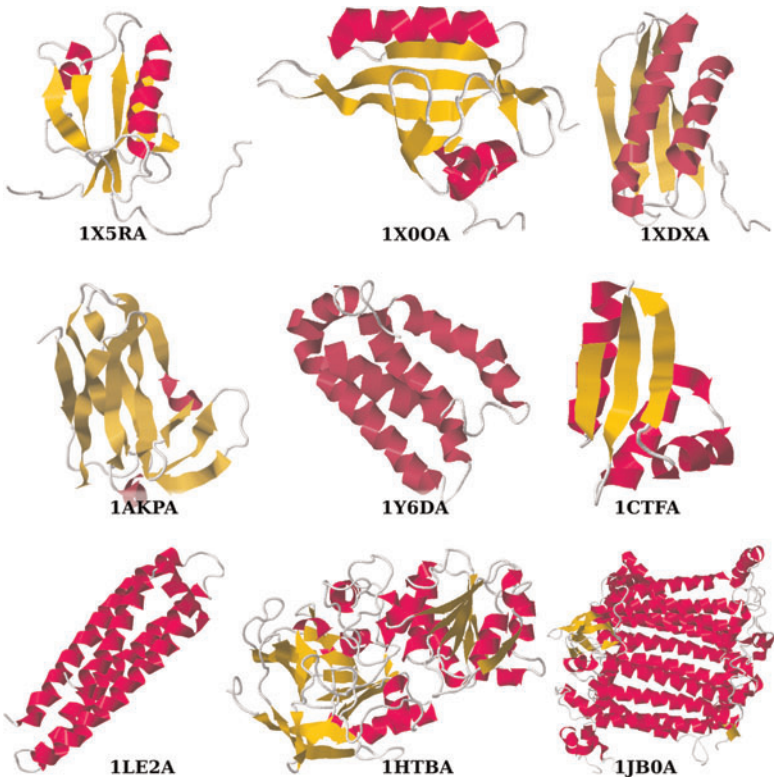


FIG. 12. Chains in the two data sets displayed using Jmol (JMol, 2010). The first five chains have almost the same length, but different distributions of secondary structures. The last four chains are from Lotan et al. (2004) and have varying lengths.

- N_P is the number of primitive pairs (bonds) tested for overlap,
- C_P is the cost of testing a primitive pair for overlap.

The three costs—C_V, C_U, and C_P—shown in Table 2 were determined experimentally. Using the chain tree, a 1° rotation of every non-locked bond (all peptide bonds and all bonds in secondary structures were locked) was carried out. The subsequent clash detection typically involved many volume intersection tests. C_V was determined by measuring the avarage CPU-time for all the resulting volume intersection tests (no matter their outcome). The transformation of one bounding volume into the coordinate system of another bounding volume was included in C_V. Rotations that did not result in clashes were used to update the chain tree. Such an update requires updating of all bounding volumes associated with nodes between the rotated bond and the root of the chain tree. C_U was determined by measuring the average CPU-time of all such bounding volume updates. The transformation of the bounding volume of the right subtree to the coordinate system of the bounding volume of the left subtree was included in C_U. To increase the accuracy of C_V and C_U, they were averaged over 100 repetitions.

Testing a pair of primitives corresponds to finding the distance between two atoms. To determine C_P, we therefore generated 10^6 random point-pairs and measured the average time it took to transform one point to another point's coordinate system, and then to find their distance.

8.3. To adjust or not to adjust

In the experiments reported in this subsection, all peptide bonds were locked. The ϕ torsion angle on the first N-C_α bond and the ψ torsion angle on the last C_α-C bond do not affect the structures of chains, so the corresponding bonds were also locked.

TABLE 2. AVERAGE COSTS (IN MS) OF CHECKING LSSS FOR OVERLAP (C_V), UPDATING AN LSS (C_U), AND CHECKING PRIMITIVES FOR OVERLAP C_P

C_V	C_U	C_P
0.00075	0.0020	0.00040

Three sets of experiments were carried out. In the first set, chain trees were set up to represent native structures of the selected chains. The coordinates of all atoms were obtained from the Protein Data Bank (PDB) (Berman et al., 2000). Ten rotations by the angles $\pm i \times 4^\circ$, $i = 1, 2, \ldots, 5$, were applied to each unlocked bond. For each selected chain, six types of chain trees were examined: unlocked or locked peptide plane, unlocked or locked (and regrouped) secondary structures, and unbalances or balanced secondary structures (only in case of regrouping).

The average costs of the first set of experiments (rotations applied to the native structures) are shown in Table 3. It is clear that grouping peptide planes, locking, grouping and balancing secondary structures provides a substantial speed-up. Also, not surprisingly, the cost improvements tend to increase for chains with high fraction of secondary structures.

The second section in Table 3 shows the average costs of clashing rotations, while the third section shows the average costs of non-clashing rotations. Ratios of clashing and non-clashing rotations are also provided. Figure 13 summarizes graphically the results of Table 3.

In the second set of experiments, adjustable chain trees were used on structures not as tight packed as was the case for native structures. This is a very typical situation during the folding process. These loosely packed structures were obtained as follows. Their initial conformation were unfolded structures: all ϕ and ψ angles were set to 180°. ω-angles of peptide bonds were extracted from the PDB. Non-clashing rotations that minimized the RMSD between the current and the native conformation were applied as long as the RMSD was above 10Å. Then, 10 rotations by the angles $\pm i \times 4^\circ$, $i = 1, 2, \ldots, 5$, were applied to each unlocked bond. Table 4 shows the average rotation costs analogous to the results for native structures

Table 3. Average Costs (in ms) of Rotations of Native Structures

Native	*1X5RA*	*1X0OA*	*1XDXA*	*1AKPA*	*1Y6DA*
All rotations					
—	0.291	0.283	0.234	0.234	0.251
P	0.226	0.197	0.164	0.213	0.164
S	0.245	0.233	0.170	0.204	0.113
PS	0.208	0.200	0.155	**0.183**	0.099
SB	0.243	0.256	0.184	0.217	0.095
PSB	**0.200**	**0.194**	**0.141**	0.188	**0.091**
Clashing rotations only					
Ratio	*33.3%*	*19.1%*	*29.4%*	*36.2%*	*15.6%*
—	0.296	0.202	0.185	0.214	0.323
P	0.209	0.165	0.148	0.180	0.134
S	0.258	0.184	0.142	0.204	0.170
PS	0.204	**0.163**	0.127	**0.160**	0.122
SB	0.260	0.203	0.160	0.178	0.131
PSB	**0.194**	0.164	**0.120**	0.164	**0.099**
Non-clashing rotations only					
Ratio	*66.7%*	*80.9%*	*70.6%*	*63.8%*	*84.4%*
—	0.288	0.309	0.257	0.244	0.242
P	0.235	0.203	0.178	0.234	0.170
S	0.239	0.248	0.182	0.204	0.109
PS	0.211	0.210	0.166	**0.196**	0.095
SB	0.235	0.272	0.195	0.235	0.092
PSB	**0.203**	**0.201**	**0.149**	0.201	**0.089**

The three-character code in the first column indicates which improvements were applied to the chain tree. A "P" indicates that peptide planes were grouped. The "S" indicates that secondary structures were locked and the chain tree was regrouped. Finally, a "B" indicates that the chain tree was rebalanced (only applicable with "S"). The fastest configuration is highlighted in boldface for each protein.

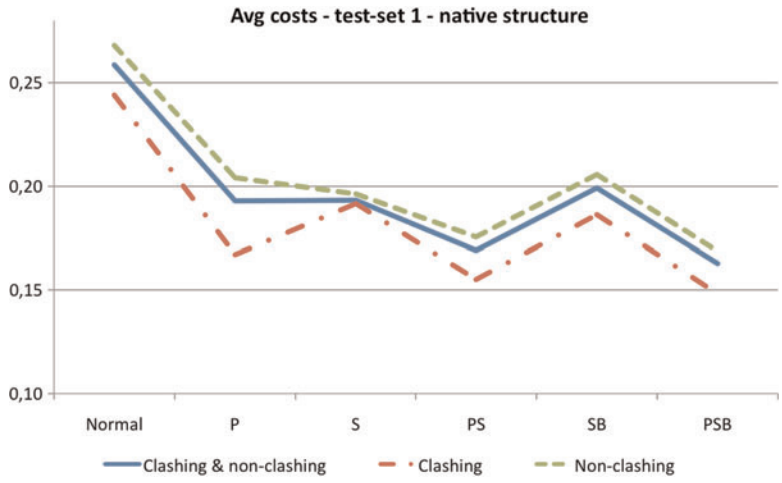


FIG. 13. Summary of Table 3. The costs are averaged over all five chains. "P" indicates that peptide planes were grouped, "S" that secondary structures were locked and grouped, and "B" that chain trees were rebalanced.

shown in Table 3. Once again, substantial speed-up can be observed when peptide planes are grouped and when secondary structures are locked, regrouped, and rebalanced. This applies equally well for clashing and non-clashing rotations. Figure 14 summarizes graphically the results of Table 4.

In the first and second sets of experiments, adjustable chain trees were intentionally tested on chains with comparable number of residues but with various fractions of secondary structures. But the usability of

TABLE 4. AVERAGE COSTS (IN MS) OF ROTATIONS OF FOLDING STRUCTURES

Folding RMSD	*1X5RA 10.0Å*	*1X0OA 10.0Å*	*1XDXA 10.0Å*	*1AKPA 10.0Å*	*1Y6DA 10.0Å*
All rotations					
—	0.223	0.241	0.158	0.231	0.147
P	0.173	0.175	0.137	0.193	0.102
S	0.190	0.206	0.122	0.184	0.110
PS	0.173	0.161	0.109	**0.172**	0.098
SB	0.202	0.212	0.137	0.200	0.110
PSB	**0.162**	**0.152**	**0.105**	0.179	**0.086**
Clashing rotations					
Ratio	*37.4%*	*31.1%*	*43.0%*	*68.2%*	*17.5%*
—	0.219	0.185	0.147	0.241	0.138
P	0.173	0.138	0.135	0.176	**0.102**
S	0.196	0.152	0.118	0.192	0.110
PS	0.183	**0.121**	0.110	**0.164**	0.123
SB	0.213	0.153	0.136	0.192	0.103
PSB	**0.171**	0.130	**0.107**	0.171	0.106
Non-clashing rotations					
Ratio	*62.6%*	*68.9%*	*57.0%*	*31.8%*	*82.5%*
—	0.225	0.295	0.170	0.212	0.149
P	0.173	0.199	0.140	0.236	0.101
S	0.186	0.266	0.127	**0.164**	0.110
PS	0.168	0.192	0.108	0.185	0.094
SB	0.195	0.277	0.137	0.211	0.112
PSB	**0.157**	**0.162**	**0.104**	0.194	**0.082**

"P" indicates that peptide planes were grouped, "S" that secondary structures were locked and grouped, and "B" that chain trees were rebalanced. The fastest configuration is highlighted in boldface for each protein.

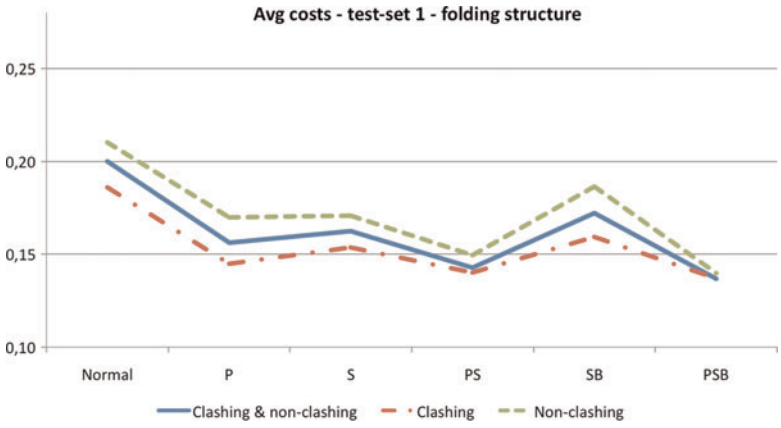


FIG. 14. Summary of Table 4. The costs are averaged over all five chains. "P" indicates that peptide planes were grouped, "S" that secondary structures were locked and grouped, and "B" that chain trees were rebalanced.

adjustable chain trees should increase as the number of residues in chains increases. In order to show this, we tested adjustable chain trees on four chains used by Lotan et al. (2004). These were 1CTFA (68 residues), 1LE2A (144 residues), 1HTBA (374 residues), and 1JB0A (755 residues).

The performance of adjustable chain trees was tested in a slightly different way than in the previous two sets of experiments, since non-clashing rotations become rare for longer chains already when the RMSD is around 30Å. If a rotation did not result in a clash and the RMSD between current and native conformation was reduced, then the next rotation was applied to the improved conformation. If the RMSD did not decrease or if the rotation resulted in a clash, the new conformation was rejected and the next rotation was applied to the old conformation. The rotations were applied until 100 consecutive rotations did not give any RMSD improvement. Structures obtained in this manner were not as tightly packed as the native structures. Hence, they can be considered as realistic structures somewhere in the middle of the folding process. Average costs of rotations in native and folding structures are reported in Table 5. Once again, a dramatic improvement can be observed. But it seems that it is not dependent on the number of residues but rather on the size of the fractions of secondary structures. Figure 15 summarizes graphically the results of Table 5.

TABLE 5. AVERAGE COSTS (IN MS) OF ROTATIONS OF STRUCTURES FROM LOTAN ET AL. (2004)

Native				
PDB-id	*1CTFA*	*1LE2A*	*1HTBA*	*1JB0A*
—	0.177	0.311	0.961	0.975
P	0.142	0.257	0.712	0.810
S	0.124	**0.101**	0.785	0.634
PS	0.101	0.113	0.605	0.608
SB	0.119	0.118	0.754	0.634
PSB	**0.087**	0.113	**0.601**	**0.593**
Folding				
RMSD	*10.9Å*	*6.3Å*	*28.9Å*	*35.2Å*
—	0.102	0.236	0.407	0.575
P	0.091	0.274	0.324	0.491
S	**0.076**	**0.103**	0.363	0.431
PS	0.082	0.123	0.301	0.394
SB	0.080	0.127	0.368	0.385
PSB	0.085	0.141	**0.299**	**0.358**

"P" indicates that peptide planes were grouped, "S" that secondary structures were locked and grouped, and "B" that chain trees were rebalanced. The fastest configuration is highlighted in boldface for each protein.

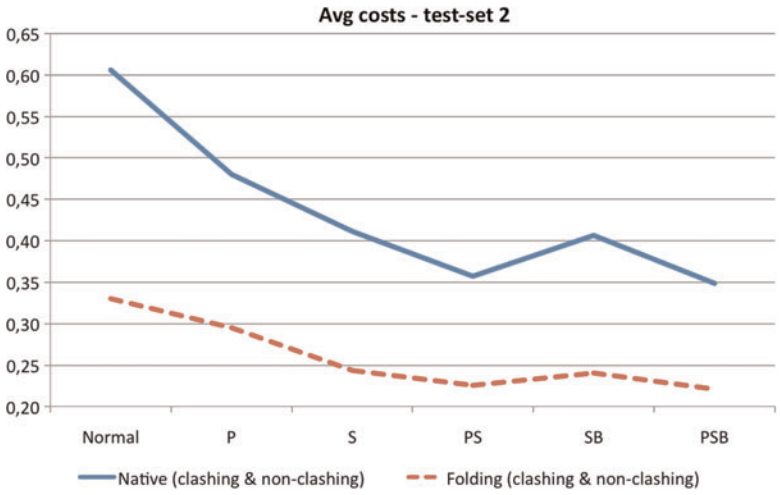


FIG. 15. Summary of Table 5. The costs are averaged over all four proteins. "P" indicates that peptide planes were grouped, "S" that secondary structures were locked and grouped, and "B" that chain trees were rebalanced.

It should be expected that the addition of side chains will reduce the speed-up obtained by using adjustable chain trees rather than standard chain trees. Neither α-helices nor β-strands with side chains can be bounded as tight. Although our implementation has not yet been extended to include side chains, we investigated the speed-up of adjustable chain trees when all bounding volumes had an added radius of 2Å, 4Å, and 6Å. Bounding volumes will then contain larger and larger portions of side chains. While the speed-up becomes less dramatic as the radius increases, it is still advantageous to use adjustable chain trees (results are not included here).

Blowing up the radii of all atoms is of course a very primitive approach. Once side chains can be bounded by their own tighter bounding volumes, the advantage of using adjustable chain trees will be restored (although it will never be as good as when side chains are ignored). In some applications, non-clashing conformations of the backbone are generated first and appropriate rotamers are added afterwards. The use of adjustable chain trees rather than standard chain trees is then an obvious choice.

9. CONCLUSION

In this article, we suggested a modification of chain trees particularly suitable for the detection of clashes in folding simulations or structure predictions of the backbone of protein chains. As α-helices and β-strands are either predicted beforehand or are created relatively early in the folding or prediction process, locking, rearranging, and rebalancing can be done in a preprocessing phase. Rearrangement of secondary structures and peptide planes results in chain trees with much tighter bounding volumes. Our results clearly indicate that adjustable chain trees provide a substantial speed-up in the detection of clashes and in the update of conformations.

We investigated elsewhere (Fonseca and Winter, 2010) how adjustable chain trees perform when other bounding volumes—such as spheres, capsules, and rectangular swept spheres—are used. Similar speed-ups were observed for all these bounding volumes when switching from the ordinary chain tree to the adjustable version.

Adjustable chain trees can also be used for the efficient determination of free energy changes when moving from one conformation to another. Bounding volumes will in this context typically have larger volumes, and more clash checks will be required. This is not only due to the increased volumes, but also to the fact that the entire chain tree has to be checked for overlaps. We believe that adjustable chain trees will prove just as useful in energy calculations as they did in clash detection.

ACKNOWLEDGMENT

We would like to thank Kevin Karplus for valuable comments and suggestions during the preparation of this manuscript.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Berman, H.M., Westbrook, J., Feng, Z., et al. 2000. The Protein Data Bank. *Nucleic Acids Res.* 28, 235–242. Available at: www.pdb.org. Accessed February 1, 2011.

Bereg, S. 2004. Cylindrical hierarchy for deforming necklaces. *Int. J. Comput. Geom. Appl.* 14, 3–17.

Branden, C., and Tooze, 1999. J. *Introduction to Protein Structure,* 2nd ed. Garland Publishing, New York.

Cormen, T.H., Leiserson, C.E., Rivest, R., et al. 2009. *Introduction to Algorithms,* 3rd ed. MIT Press, Cambridge, MA.

Eberly, D.H. 2000 *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann, New York.

Ericson, C. 2005 *Real-Time Collision Detection*. Elsevier, New York.

Fonseca, R., and Winter, P. 2010. Bounding volumes for proteins—a comparative study [Technical Report]. Department of Computer Science, University of Copenhagen, Denmark.

Gottschalk, S., Lin, M., and Manocha, D. 1996. OBB-Tree: a hierarchical structure for rapid interference detection. *Proc. 23rd Annu. Conf. Comput. Graphics Interactive Techniq.* 171–180.

JMol. 2010. JMol: an open-source Java viewer for chemical structures in 3D. Available at: www.jmol.org. Accessed February 1, 2011.

Kolodny, R., Guibas, L., Levitt, M., et al. 2005. Inverse kinematics in biology: the protein loop closure problem. *Int. J. Robot. Res.* 24, 151–163.

Larsen, E., Gottschalk, S., Lin, M.C., et al. 2000. Fast distance queries with rectangular swept sphere volumes. *Proc. IEEE Conf. Robot. Automation* 4, 3719–3726.

Lotan, I., Schwarzer, F., Halperin, D., et al. 2004. Algorithm and data structures for efficient energy maintenance during Monte Carlo simulation of proteins. *J. Comput. Biol.* 11, 902–932.

Wikipedia. 2010. File:Ramachandran plot general 100K.jpg. Available at: http://en.wikipedia.org/wiki/File:Ramachandran_plot_general_100K.jpg. Accessed February 1, 2011. Lovell, S.C., Davis, I.W., Arendall III, W.B., et al. Structure validation by Cα geometry: ϕ, ψ and Cβ deviation: Proteins, Structure, Function, and Bioinformatics, Vol. 50, Issue 3, pages 437–450, February 15, 2003.

Address correspondence to:
Dr. Pawel Winter
Department of Computer Science
University of Copenhagen
Universitetsparken 1
2100 Copenhagen O, Denmark

E-mail: pawel@diku.dk

5.6 Bounding Volumes for Proteins: A Comparative Study

The following 15 pages contains the accepted version of our paper "Bounding Volumes for Proteins: A Comparative Study. Accepted: Journal of Computational Biology. 2012" [62].

Bounding Volumes for Proteins - A Comparative Study

Rasmus Fonseca and Pawel Winter*

Abstract

A chain tree is a data structure for representing changing protein conformations. It enables very fast detection of clashes and free potential energy calculations. The efficiency of chain trees is closely related to the bounding volumes associated with chain tree nodes. A protein subchain associated with a node of a chain tree will clash with another subchain only if their bounding volumes intersect. It is therefore essential that bounding volumes are as tight as possible while intersection tests can be carried out efficiently. We compare the performance of four different types of bounding volumes in connection with the rotation of protein bonds. It is observed that oriented bounding boxes are not as good as could be expected judging by their extensive use in various applications. Both rectangular- and line swept-spheres are shown to have very good tightness of fit but the line-swept, or even simple spheres, are shown to be significantly faster because of quick overlap checks. We also investigate how the performance of the recently introduced adjustable chain trees is affected by different bounding volume types.

1 Introduction

A chain tree is a data structure for fast clash detection and free energy maintenance of folding protein chains. Suppose that during the simulation of the folding process, an attempt to rotate around a backbone or a side chain bond is made. Such a rotation should be undone if it results in a clash (either during the rotation or at the end of the rotation). Clashing rotations have to be identified and rejected as quickly as possible. If a rotation is not clashing, the free energy of the new configuration has to be calculated. This estimate is crucial for the decision if the rotation should be accepted or rejected (note that in many methods the increase of the free energy does not always cause a rejection).

The chain tree is a binary tree where each node is a subchain of atoms. Each node is furthermore associated with a transformation matrix and a bounding volume. To maintain the conformation of a protein chain during folding, a brute-force method can be employed that requires $\Theta(n)$ update-time for each conformational change and $\Theta(n^2)$-time to detect

*Department of Computer Science, University of Copenhagen, Universitetsparken 1, 2100 Copenhagen O, Denmark, e-mail: pawel@diku.dk, rfonseca@diku.dk

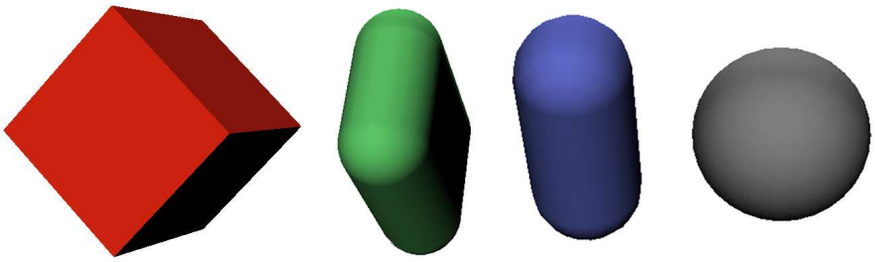


Figure 1: The four bounding volume types compared: Oriented bounding box (OBB), rectangular swept sphere (RSS), line-segment swept sphere (LSS) and sphere (PSS).

clashes. Lotan et al. (2004) showed that using a chain tree this is reduced to $\Theta(\log n)$ update time and $\Theta(n^{4/3})$ clash-detection time.

We previously suggested (Winter and Fonseca (2011)) a modification of chain trees based on the assumption that portions of folding proteins (such as α-helices and β-strands) are formed relatively early in the folding process and they remain stable throughout many (if not all) iterations of the simulation. Bonds of such subchains can be locked and their bounding volumes therefore remain unchanged. As a consequence, the chain tree can be rearranged (so that locked subchains are tightly covered by a single interior node of the chain tree) and rebalanced. The adjustable chain tree has an improved running-time compared to the standard chain tree and the asymptotic running-time stays unchanged.

We study four common bounding volumes that are reasonable for clash detection in protein chains. These are: oriented bounding boxes, rectangular swept spheres, line swept spheres and spheres (Figure 1). The computational results indicate that spheres, with their trivial intersection test and simple update methods, perform equally well as line swept spheres that have a better tightness of fit. Oriented bounding boxes and rectangular swept spheres are clearly inferior. This is particularly surprising as oriented bounding boxes have been a standard choice for this type of application. As chain trees are adjusted to the protein to fit peptide planes and secondary structures more tightly, line swept spheres become slightly more efficient than spheres while oriented bounding boxes and rectangular swept spheres remain inferior.

This paper is organized as follows. A brief description of proteins, chain trees and adjustable chain trees is given in Section 2. The different types of bounding volumes are discussed in Section 3. Computational result are given in Section 4 and, finally, concluding remarks and suggestions for further research are collected in Section 5.

2 Protein chain trees

A protein is an organic compound made of n amino acids arranged in a linear chain. A given sequence of amino acids always folds into the same 3-dimensional structure referred to as the *native conformation* of that sequence. The lengths and angles between adjacent atoms in the

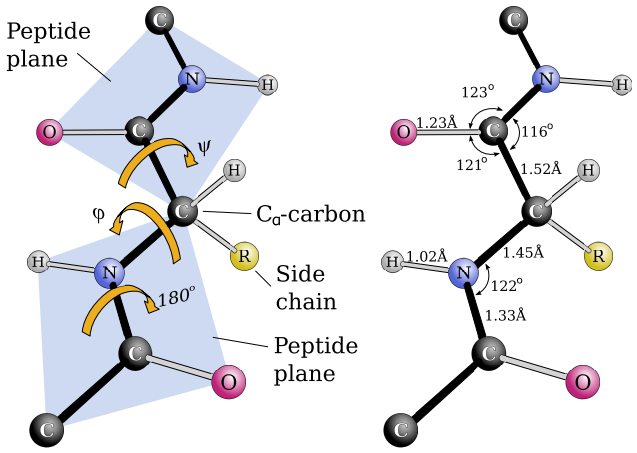


Figure 2: Backbone geometry and definitions. (left) Since the C-N bond can be fixed at 180°, peptide planes containing 6 backbone atoms each are formed. The angles around the N-C_α and C_α-C bonds, (Φ, Ψ) are the torsion angles of the chain. (right) Typical backbone bond lengths and angles

chain are mostly fixed due to steric interactions as shown in Figure 2, so the conformation of the entire protein can be completely defined by the torsions around the covalent bonds.

Protein structure prediction is the problem of predicting the native conformation. This is typically done using Monte Carlo simulations or other metaheuristics where a conformation is iteratively improved by changing one or more torsion angles slightly.

After each rotation around a bond a check is performed to ensure that no two atoms clash. A brute-force method to do this is to first transform all atoms succeding the bond, which takes $\mathcal{O}(n)$ time. The pairwise distance from all atoms preceding the bond to all atoms succeding it are then measured, which takes $\mathcal{O}(n^2)$ time. This is much too inefficient for long protein chains.

The chain tree (Lotan et al. (2004)) is a data structure for maintaining the conformation of a chain which enables $\mathcal{O}(\log n)$ update time and $\mathcal{O}(n^{1.5})$ time to check for clashes, assuming the chain is well-behaved (i.e. not extremely self-colliding). The chain tree is a balanced binary tree where leaves correspond to bonds in the protein chain and internal nodes correspond to a sub-chain. Each internal node is associated with a transformation matrix and a volume bounding all the leaves in its subtree. The transformation matrices are used to maintain fast updates. Since two non-overlapping bounding volumes exclude a clash of their respective sub-chains, the bounding volumes are used to maintain fast collision checks. For a detailed description of chain trees see Lotan et al. (2004).

An adjustable chain tree (Winter and Fonseca (2011)) is a modification of the standard chain tree. In adjustable chain trees, bonds in secondary structures, such as α-helices and β-strands, are grouped so that they have the same lowest common ancestor. As a result of this grouping, adjustable chain trees are rebalanced (regarding roots as nodes of height 0) and the bounding volumes are recomputed to tightly fit their underlying atoms (not just their two children). Bounding volumes associated with nodes of secondary structures will be

much tighter as these structures fit very well into most bounding volumes.

Adjustable chain trees are also adjusted so that every three bonds of each peptide plane have a common lowest ancestor. This results in very tight bounding volumes at the lowest levels of the chain tree. A detailed description of adjustable chain trees, as well as computational results documenting the speed-up, can be found in Winter and Fonseca (2011).

3 Bounding Volumes

There are four operations that are required to use a particular type of bounding volume in the chain tree:

- transform a bounding volume to another coordinate system,
- decide if two bounding volumes intersect,
- find a tight volume bounding two smaller volumes and
- find a tight volume bounding a set of bonds or atoms.

Assuming that all backbone atoms have the same radii, the fourth operation can be reduced to finding the minimum volume bounding a set of points. The following subsections discuss how these operations are performed for four different types of bounding volumes: oriented bounding boxes, rectangular swept spheres, line-segment swept spheres, and point swept spheres.

3.1 Oriented Bounding Boxes

An *oriented bounding box* (OBB) is a rectangular block with an arbitrary orientation (Gottschalk et al. (1996)). There are many possible representations of OBBs. The center point, an orthonormal set of three orientation vectors and three numbers indicating the extents of the box have been used in this study.

To transform a point from one coordinate system to another, both a translation and a rotation of the point is necessary. For vectors, however, only the rotation should be performed. The coordinate transformation of an OBB is obtained by transforming (rotating and translating) the center point and rotating the orientation vectors.

To decide if two OBBs B_l and B_r in 3D intersect, it is not enough to check if the vertices of B_l are all on the outside of B_r, and vice versa. Instead it is observed that B_l and B_r are disjoint if there exists a plane that separates B_l and B_r. The projections of B_l and B_r onto a line (axis) orthogonal to the separating plane will correspond to two disjoint line-segments if there is no overlap. It can be shown that it is enough to check 15 such axes: 6 corresponding to all orientations of B_l and B_r and 9 defined by pairs of orientation vectors, one from B_l and one from B_r respectively (Ericson (2005)).

OBBs bounding a point sets can be determined using principal components analysis (PCA) as discussed in Ericson (2005). The three principal component vectors are used as the orientation of the box. The center and extents can then easily be defined so the box is as small as possible given the orientation.

The OBB bounding two smaller OBBs is determined using the above method applied to the 16 corners of a given pair of OBBs. There are several methods to improve the tightness of a bounding OBB, such as using PCA only on the convex hull of the points. We have not implemented these improvements.

3.2 Rectangular Swept Spheres

A *rectangular swept sphere* (RSS) is defined as the Minkowski sum of an arbitrarily oriented flat rectangle and a sphere (Larsen et al. (2000); Eberly (2000)). The rectangle is represented by a center point, two orthogonal vectors (defining the orientation) and two numbers specifying the extents of the rectangle.

A RSS is transformed into another coordinate system in a similar way as OBBs are.

The intersection test between two RSSs is as described in Larsen et al. (2000) and amounts to computing the distance between their defining rectangles R_l and R_r in 3D. Assume first that the closest pair of points $p_l \in R_l$ and $p_r \in R_r$ lie on the edges of R_l and R_r. For each pair of edges, $e_{li} \in R_l$ and $e_{rj} \in R_r$, determine the closest pair of points p_{li} and p_{rj}. For a given i and j, iff the following two conditions are met, then $p_l = p_{li}$ and $p_r = p_{rj}$.

- The open upper half-space bounded by the plane through p_{li} with $\overrightarrow{p_{li}p_{rj}}$ as normal contains none of the corners in R_l.
- The open upper half-space bounded by the plane through p_{rj} with $\overrightarrow{p_{rj}p_{li}}$ as normal contains none of the corners in R_r.

If these conditions are not met for any pair of edges, then either p_l or p_r are in the interior of one of the rectangles. In this case, the distance between R_l and R_r is the largest of the following two values: The separation between R_l and R_r projected onto an axis normal to R_l or projected onto an axis normal to R_r.

A method for determining an RSS bounding a point set is described in Larsen et al. (2000). First, the principal components, $\overrightarrow{v_0}, \overrightarrow{v_1}$ and $\overrightarrow{v_2}$, of the point set are determined. The two components corresponding to the largest spread, $\overrightarrow{v_0}$ and $\overrightarrow{v_1}$ fix the orientation of the rectangle. Next, the thinnest slab normal to $\overrightarrow{v_2}$ that encloses all the points is determined. The width of this slab is the diameter of the RSS, and the mid-plane, i.e. the set of points equally far from the slab boundaries, contains the mid-point of the rectangle. The mid-point and the extents are adjusted such that the points projected onto a plane normal to $\overrightarrow{v_0}$ and another plane normal to $\overrightarrow{v_1}$ are enclosed in 2D capped cylinders. Even though all points now appear to be inside the RSS when looking along $\overrightarrow{v_0}$ and $\overrightarrow{v_1}$ they might still fall outside near the capped corners of the RSS. If this is the case, the rectangle of the RSS is extended outward at a 45° angle until all points are enclosed.

A RSS bounding two RSSs with radii r_l and r_r is obtained in the following way. First a RSS bounding the eight corner points of the two rectangles is constructed. Its radius is then extended by $\max(r_l, r_r)$.

3.3 Line-Segment Swept Spheres

A *line-segment swept sphere* (LSS) is the Minkowski sum of an arbitrarily oriented line-segment and a sphere. A LSS is therefore fully specified by the two end-points of the line-segment and the radius of the sphere. A LSS is also sometimes referred to as a line swept sphere, a capsule, a capped cylinder, a spherocylinder (Ericson (2005)) or even as a cigar (Bereg (2004)).

A LSS is transformed to another coordinate system by applying the appropriate transformation matrix to the end-points of its defining line-segment.

Two LSSs intersect if the shortest distance between their defining line-segments is shorter than their combined radii. A very fast method to find the distance between two line-segments is described in Ericson (2005).

To find a good LSS bounding a set of points, the first principal component, $\overrightarrow{v_0}$, of the points is used as the direction of the defining line-segment. The points are projected onto a plane normal to $\overrightarrow{v_0}$ and the smallest enclosing circle C with radius r is found using Welzl's algorithm (Welzl (1991)). C together with $\overrightarrow{v_0}$, defines an infinitely extended cylinder that contains all points. Finally, two hemispheres, both with radius r, are "slid" along the infinite line-segment from each side until their surfaces touch the bounded points. The centers of these hemispheres are end-points of the defining line-segment.

To find a good LSS, B, bounding two smaller LSSs B_l and B_r, the four spheres defined by the end-points of the line-segments of B_l and B_r and their respective radii are considered. The direction $\overrightarrow{d}$ of B is determined as the direction between the two most remote points on the surfaces of these four spheres. The four spheres are projected onto a plane orthogonal to $\overrightarrow{d}$ as four circles (one will be nested in another and can be disregarded). The smallest circle enclosing the remaining three circles (the problem of Apollonius) together with $\overrightarrow{d}$, defines an infinitely extended cylinder containing all four spheres. Finally, two hemispheres are "slid" in place, as explained above.

3.4 Point Swept Spheres

A *point swept sphere* (PSS) is simply a sphere. We use the term PSS to emphasize its relation to the other two bounding volumes introduced above and to have a meaningful abbreviation. A PSS is represented by its center and radius.

The coordinate transformation is done simply by transforming the center-point.

Two PSSs intersect if the distance between their centers is less than their combined radii. Finding the PSS bounding two PSSs is trivial. See e.g. Ericson (2005) for further details. Welzl's algorithm (Welzl (1991)) gives an expected linear time algorithm for finding the minimum radius sphere bounding a set of points. Note that, contrary to the other three volume types, the PSS is the only one that guarantees a minimum bounding volume.

4 Computational Results

The improvement in efficiency of the adjustable chain tree over the standard chain tree was documented in (Winter and Fonseca (2011)). This paper focuses on analyzing which bounding volume gives the best efficiency, both in the standard and the adjustable chain tree. First, the data set used for the different experiments is described. Second, a cost measure is defined which can be used to evaluate the speed-up independently of implementational and hardware details. Finally we show to what extent different bounding volume types speed up clash-detection.

4.1 Data Set

The same data set as in Winter and Fonseca (2011) is used. The first five chains have a similar length of 110-119 residues, the typical length in PDBSelect 25 (Berman et al. (2000)). They are selected so that they have different fractions of residues in secondary structures (See Table 1). The last four chains are from the seminal paper on chain trees (Lotan et al. (2004)). The lengths vary from 68 to 755 residues. Figure 3 shows all nine proteins.

4.2 Cost Measure

To evaluate the efficiency of a particular bounding volume independent of implementational and hardware issues, we determine the average *cost* of a single rotation in the chain tree. The cost of a rotation is defined as

$$N_V C_V + N_U C_U + N_P C_P$$

where

- N_V is the number of bounding volume pairs tested for overlap,
- C_V is the cost of testing two bounding volumes for overlap,
- N_U is the number of bounding volumes updated due to rotations,
- C_U is the cost of updating a bounding volume,
- N_P is the number of primitive pairs (bonds) tested for overlap,
- C_P is the cost of testing a primitive pair for overlap.

The three costs, C_V, C_U and C_P shown in Table 2, were determined experimentally from the chain trees of the native conformations in the data set. A 1° rotation of every non-locked bond (all peptide bonds and all bonds in secondary structures were locked) was carried out. The subsequent clash detection typically involved many volume intersection tests. C_V was determined by measuring the average CPU-time for all volume intersection tests (no matter their outcome). The transformation of one bounding volume into the coordinate system of another was included in C_V. Rotations that did not result in clashes were used to update the chain tree. Such an update requires updating all bounding volumes associated with nodes between the rotated bond and the root of the chain tree. C_U was determined by measuring

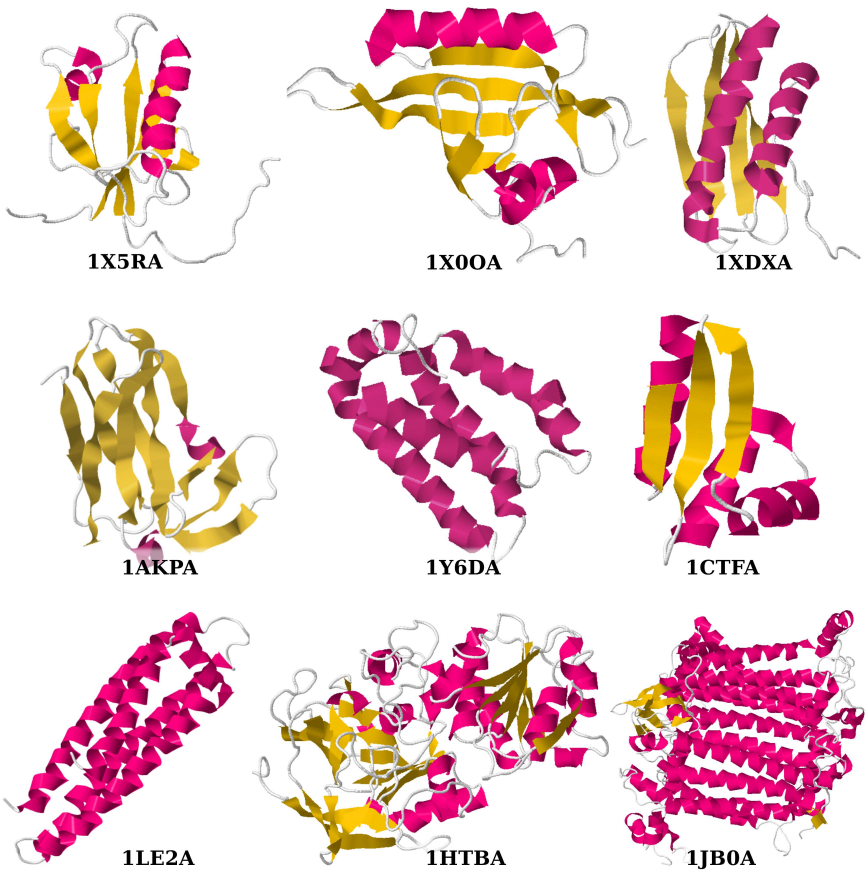


Figure 3: Chains in the two data sets. The first five chains have almost the same length, but different distributions of secondary structures. The last four chains are from Lotan et al. (2004) and have varying lengths.

the average CPU-time for all such bounding volume updates. The transformation of the bounding volume of the right subtree to the coordinate system of the bounding volume of the left subtree was included in C_U. To increase the accuracy of C_V and C_U, their computation times were averages of 100 repetitions.

Testing a pair of primitives corresponds to finding the distance between two atoms. To determine C_P, we therefore generated 10^6 random point-pairs and measured the average time it took to transform one point to another point's coordinate system and find their distance.

The number of bounding volume pairs tested for overlap, N_V, will be a good indicator of a volumes 'tightness of fit'. However, to measure the tightness of fit explicitly we introduce the *relative volume difference* between two volume types. Given an adjustable chain tree representing a protein structure, the relative volume difference between two bounding volume types, BV_1 and BV_2, is defined as

$$RVD(BV_1, BV_2) = \frac{1}{N} \sum_n \frac{V_{BV_1}(n) - V_{BV_2}(n)}{V_{BV_2}(n)}$$

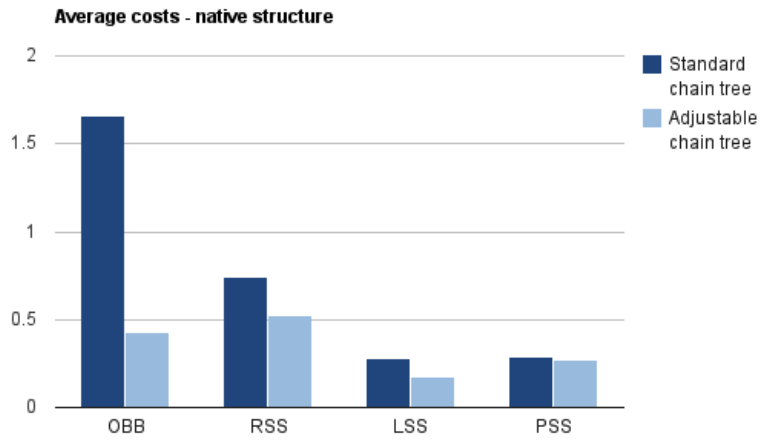


Figure 4: Average costs (in ms) of a rotation in a standard chain tree and an adjustable chain tree. These measurements are performed on the native conformation which is typically as compact as possible for a protein.

where the sum is over all nodes (interior and leaves) in the adjustable chain tree (N) and $V_{BV}(n)$ is the volume of the bounding volume associated with node n.

4.3 Comparison of Bounding Volumes

In all experiments reported in this subsection, peptide bonds were locked in the adjustable chain tree. The Φ torsion angle on the first N-C_α bond and the Ψ torsion angle on the last C_α-C bond do not affect the conformation of the protein, so the corresponding bonds were therefore also locked in the adjustable chain trees. Bonds in secondary structure segments were also locked.

Two sets of experiments were carried out. In the first set, both standard chain trees and adjustable chain trees were set up to represent native conformations of the selected chains. The coordinates of all atoms were obtained from the Protein Data Bank (PDB) (Berman et al. (2000)). For each of the 9 chains, ten rotations by the angles $\pm i \times 4°, i = 1, 2, ..., 5$, were applied to each unlocked bond. The average rotation costs (including the intersection tests) for each of the four types of bounding volumes are shown in Figure 4 (both in standard chain trees and in adjustable chain trees). Tables showing the measurements in numbers can be found in the appendix.

It is clear that the adjustable chain trees with grouped peptide planes and secondary structures provide a substantial speed-up when using both OBBs, RSSs and LSSs. Surprisingly, this speed-up is not observed for PSSs. The reason for this is that spheres have no elongation. Their tightness of fit therefore does not improve when the adjustable chain trees group long segments of α-helix and β-strand. However, for the standard chain trees, spheres are one of the fastest bounding volume types together with LSSs. This indicates that the results in the seminal paper on chain trees (Lotan et al. (2004)) could have been improved significantly by replacing the RSSs with LSSs or PSSs.

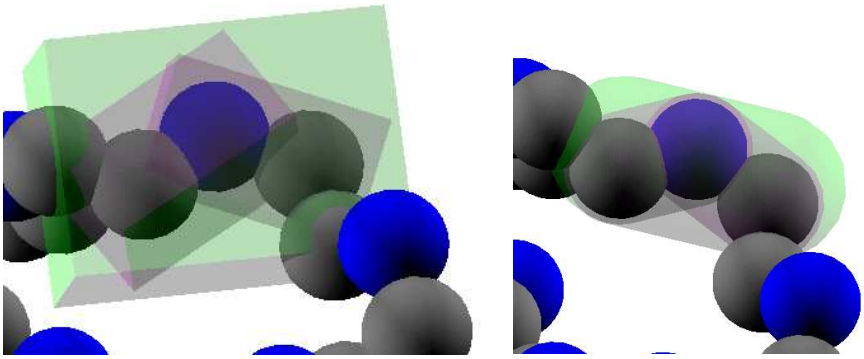


Figure 5: OBBs and LSSs of the two lowest levels in a standard chain tree. The volume of OBBs quickly increase when climbing the tree because they bound the corners of their children.

Another significant feature of Figure 4 is that the average cost of OBBs decreases dramatically when using adjustable chain trees. The reason is that OBBs have sharp corners. Therefore, in the standard chain trees, OBBs of nodes just above leaves are significantly larger than the corresponding swept sphere volumes (see Figure 5). In the adjustable chain trees the four atoms in peptide planes are collected under a single node and the bounding volumes are calculated based on atom positions and not children volumes. This improvement greatly reduces the size of OBBs in particular.

The cost measure is made up of three terms. For all volume types the main contribution to the average cost is the term $N_V C_V$. For any given rotation, N_U is rarely larger than 10 and N_P is typically less than 50. As shown in Figure 6, however, N_V, for the proteins in our dataset, is typically in the range between 200 to 500 which makes the overlap checks of bounding volumes the most time-consuming part of a rotation. From Figure 6 it is noted

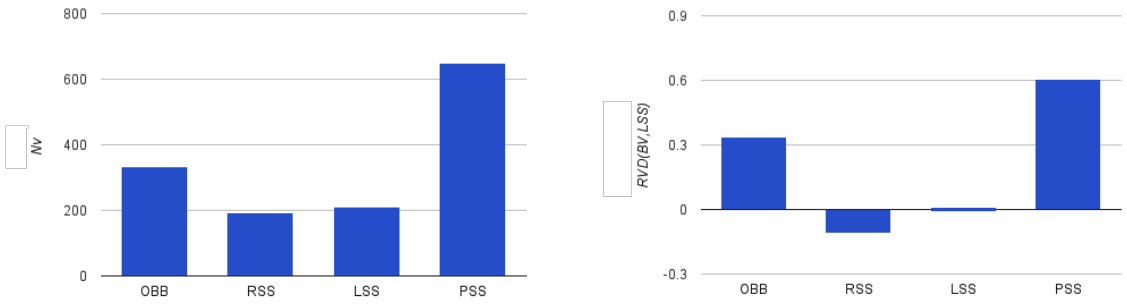


Figure 6: The tightness of fit for different bounding volume types, (left) the average number of overlap checks in a rotation, N_V and (right) the relative volume difference between a particular volume type and an LSS.

that both in terms of N_V and relative volume difference (averaged over adjustable chain trees

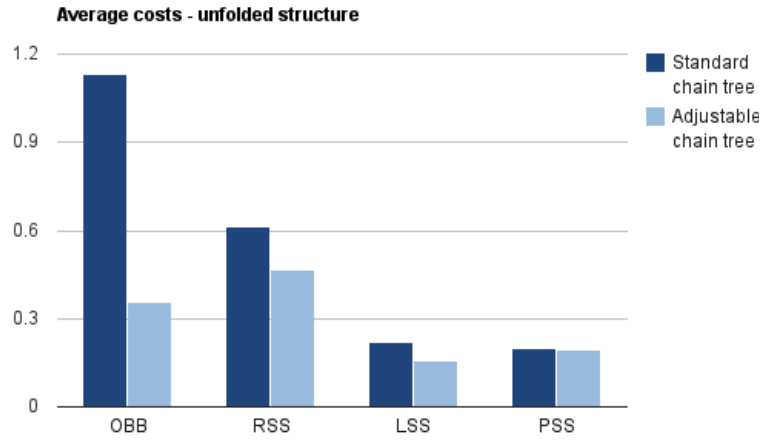


Figure 7: Average costs (in ms) of a rotation in a standard chain tree and an adjustable chain tree. These measurements are performed on a partially unfolded conformation.

for proteins specified in Section 4.1), the RSSs are the most tight-fitting bounding volumes closely followed by LSSs. Since C_V of the RSSs is 3 times that of LSSs, however, the total $C_V N_V$ term of LSSs is smaller.

As mentioned, both LSSs and PSSs perform well in the standard chain trees but LSSs are slightly faster. For adjustable chain trees the LSSs are 37% faster than PSSs, so it is safe to conclude that the tightness of fit that the LSSs have outweigh the fast overlap check of the PSSs. The main conclusion of this paper is therefore that LSSs are the optimal choice of bounding volumes for protein chain trees. We assume that this result holds true for bounding volume hierarchies of chains in general.

The native conformations, that are used as starting conformations for the experiments above, are all very tightly packed. To verify that the results hold with a more loosely packed conformation where there are less clashes after each rotation, the experiments above were repeated with a partially unfolded conformations. As shown in Figure 7, the same trend as in Figure 4 are observed. To check that the cost measure is not overly simplistic, the average CPU-time of a rotation was also measured. As shown in Figure 8 the same trend is observed again.

5 Conclusions

In this paper we compared different bounding volumes that can be associated with the nodes of chain trees. LSSs seem to perform much better than OBBs and RSSs that have been used as standard bounding volumes in other applications of chain trees to proteins. The performance of bounding volumes was shown to be a tradeoff between fast collision checks and tightness of fit. Our results clearly indicate that RSSs are the most tight-fitting volumes, but a substantial speed-up is possible when using LSSs because of their fast overlap checks. Another, perhaps somewhat surprising conclusion is that, at least for shorter chains,

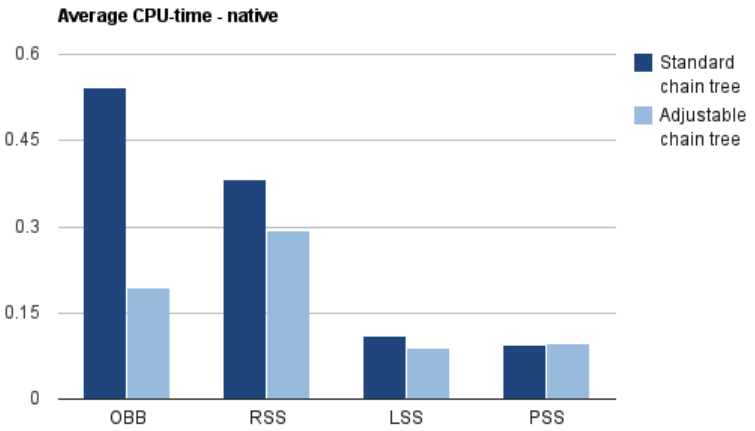


Figure 8: Average CPU-time (in ms) of a rotation in a standard chain tree and an adjustable chain tree.

PSSs perform comparable with LSSs.

Bounding volumes also play an essential role when adjustable chain trees are used for the efficient determination of free energy changes when moving from one conformation to another. Bounding volumes will in this context typically have greater radius and more intersection tests will be required. This is not only due to the increased volumes but also to the fact that entire chain tree has to be checked for overlaps. We therefore believe that adjustable chain trees and LSSs will prove even more useful in energy calculations than they did in clash detection.

References

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov and P.E. Bourne, The Protein Data Bank, Nucleic Acids Research, (2000) 28: 235-242 (www.pdb.org).

S. Bereg, Cylindrical Hierarchy for Deforming Necklaces, Int. J. Comput. Geometry Appl. 14 (2004) 3-17.

D.H. Eberly, *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*, Morgan Kaufmann (2000).

C. Ericson, *Real-Time Collision Detection*, Elsevier (2005).

S. Gottschalk, M. Lin, and D. Manocha, OBB-Tree: A Hierarchical Structure for Rapid Interference Detection, Proc. of the 23-rd Ann. Conf. on Computer Graphics and Interactive Techniques (1996) 171-180.

E. Larsen, S. Gottschak, M.C. Lin and D.Manocha, Fast Distance Queries with Rectangular Swept Sphere Volumes, Proc. of IEEE Conf. on Robotics and Automation 1:4 (2000) 3719–3726.

T. Larsson and T. Akenine-Moller, Bounding Volume Hierarchies of Slab Cut Balls, Computer Graphics Forum 28:8 (2009) 2379-2395.

I. Lotan, F. Schwarzer, D. Halperin and J.C. Latombe, Algorithm and Data Structures for Efficient Energy Maintenance During Monte Carlo Simulation of Proteins, J. Comput. Biol. 11 (2004) 902-932.

E. Rimon and S. Boyd, Efficient Distance Computation Using Best Ellipsoid Fit, Proceedings of the 1992 IEEE International Symposium on Intelligent Control (1992) 360-365.

E. Welzl, Smallest Enclosing Disks (Balls and Ellipsoids), in H. Maurer (ed.) *New Results and New Trends in Computer Science*, Lecture Notes in Computer Science 555 (1991) 359-370.

P. Winter and R. Fonseca, Adjustable Chain Trees for Proteins, J. Comput. Biol. (2011) ahead of print. doi:10.1089/cmb.2010.0320

A Tables

PDB-id	# res	% in $\alpha\beta$	$\#\alpha + \#\beta$
1X5RA	112	42%	2 + 6
1X0OA	119	58%	3 + 5
1XDXA	114	63%	2 + 4
1AKPA	114	79%	2 + 11
1Y6DA	114	82%	7 + 0
1CTFA	68	82%	3 + 3
1LE2A	144	83%	5 + 0
1HTBA	374	52%	10 + 19
1JB0A	755	62%	12 + 4

Table 1: Second column is the number of residues in the examined proteins. The third column indicates the portion of residue in secondary structures. The fourth column indicates the number of α-helices and β-strands.

	OBB	RSS	LSS	PSS
C_V	0.0011	0.0024	0.00075	0.00041
C_U	0.0083	0.0089	0.0020	0.00027
C_P	0.00040			

Table 2: Average costs (in ms) of checking bounding volumes for overlap (C_V), updating a bounding volume (C_U) and checking primitives for overlap (C_P). The cost of testing a pair of primitives does not depend on the bounding volume type.

OBB		RSS		OBB		OBB	
SCT	ACT	SCT	ACT	SCT	ACT	SCT	ACT
1.719	0.554	0.837	0.669	0.317	0.212	0.333	0.323
1.696	0.523	0.805	0.618	0.300	0.210	0.305	0.294
1.515	0.367	0.642	0.435	0.244	0.134	0.253	0.248
1.496	0.452	0.735	0.593	0.257	0.214	0.351	0.374
1.888	0.243	0.706	0.320	0.267	0.099	0.190	0.131

Table 3: Average costs (in ms) of a rotation in a standard chain tree (SCT) and an adjustable chain tree (ACT) (see also Figure 4).

OBB		RSS		OBB		OBB	
SCT	ACT	SCT	ACT	SCT	ACT	SCT	ACT
1.046	0.661	0.246	0.204	0.394	0.561	0.178	0.206
1.340	0.723	0.266	0.240	0.450	0.541	0.179	0.225
0.920	0.508	0.179	0.150	0.284	0.357	0.115	0.145
1.259	0.711	0.251	0.270	0.403	0.553	0.200	0.275
1.106	0.469	0.167	0.126	0.256	0.322	0.112	0.115

Table 4: Average costs (in ms) of a rotation in a standard chain tree (SCT) and an adjustable chain tree (ACT) (see also Figure 7). These measurements are performed on a partially unfolded conformation.

OBB		RSS		OBB		OBB	
SCT	ACT	SCT	ACT	SCT	ACT	SCT	ACT
0.556	0.414	0.124	0.112	0.235	0.351	0.102	0.119
0.555	0.403	0.117	0.098	0.226	0.323	0.102	0.099
0.495	0.354	0.097	0.081	0.175	0.260	0.079	0.091
0.501	0.381	0.108	0.120	0.199	0.324	0.100	0.125
0.604	0.355	0.107	0.064	0.140	0.211	0.060	0.051

Table 5: Average CPU-time (in ms) of a rotation in a standard chain tree (SCT) and an adjustable chain tree (ACT) (see also Figure 8).

5.7 Protein Packing Quality using Delaunay Complexes

The following 17 pages contains the published version of our paper "Protein Packing Quality using Delaunay Complexes. R. Fonseca. P. Winter and K. Karplus. Proceedings of ISVD 2011. Pages 117-122. 2011" [63].

Protein Packing Quality Using Delaunay Complexes

Rasmus Fonseca
Dept. of Computer Science
University of Copenhagen
Copenhagen, Denmark
Email: rfonseca@diku.dk

Pawel Winter
Dept. of Computer Science
University of Copenhagen
Copenhagen, Denmark
Email: pawel@diku.dk

Kevin Karplus
Biomolecular Engineering Dept.
University of California
Santa Cruz, USA
Email: karplus@soe.ucsc.edu

***Abstract*—A new method for estimating the packing quality of protein structures is presented. Atoms in high quality protein crystal structures are very uniformly distributed which is difficult to reproduce using structure prediction methods. Packing quality measures can therefore be used to assess structures of low quality and even to refine them.**

Previous methods mainly use the Voronoi cells of atoms to assess packing quality. The presented method uses only the lengths of edges in the Delaunay complex which is faster to compute since volumes of Voronoi cells are not evaluated explicitly. This is a novel application of the Delaunay complex that can improve the speed of packing quality computations. Doing so is an important step for, e.g., integrating packing measures into structure refinement methods. High- and low-resolution X-ray crystal structures were chosen to represent well- and poorly-packed structures respectively. Our results show that the developed method is correlated to the well-established RosettaHoles2 but three times faster.

***Keywords*-Delaunay complex; protein; packing quality;**

I. Introduction

The resolution of a protein structure indicates how accurate the experimentally determined positions of atoms are in the protein. Protein structures with resolutions less than 1.8Å are generally considered good and they are, paradoxically, referred to as high-resolution structures. High-resolution structures are characterized by a uniform distribution of atoms in the core. Low-resolution structures and structures solved partially or wholly by computational methods tend to form clusters of atoms in some places and holes or voids in others. This is referred to as bad packing of the atoms. An estimate of the packing quality can be used to improve structure assessment software such as WHAT-CHECK [1], PROCHECK [2] or ProSA [3]. Also, it can be added as an additional term in free energy functions used in protein structure prediction or refinement.

A number of methods have been developed to characterize packing [4], [5], many of which use the volumes of Voronoi cells for atoms [6], [7], [8]. A very recent and popular method is RosettaHoles2 [9], [10]. This method uses the Voronoi diagram and a support vector machine to output a packing energy. For each atom, twenty spheres with increasing radii are centered on the atom. The volumes of the intersections between the spheres and the Voronoi cell of the atom are used as input features to the support vector machine. The number characterizing the packing, the *RosettaHoles2 cost*, is found by averaging the output of the support vector machine for all atoms.

We use the Delaunay complex of all heavy (non-hydrogen) atoms to quantify the packing quality of protein structures. The Delaunay complex, $DC(A)$, of a set of points, A, consists of all 3-simplices (tetrahedra) whose circumsphere does not contain a point of A in its interior, as well as all faces of simplices in $DC(A)$. The 1-simplices in $DC(A)$ are a set of edges between points of A. In this study we assume that all atoms have roughly the same radii and hence can be represented by a set of points. The packing quality is found using only the edges of $DC(A)$ as input features to a feed-forward neural network. Because the faces of the Voronoi cell intersect the edges of the Delaunay complex at their midpoints, the lengths of edges roughly capture the geometry of the Voronoi cell. However, much less computation is required when the cell volume is not explicitly calculated. For training purposes, high-resolution and low-resolution structures are used to represent well-packed and poorly-packed structures respectively.

Our method distinguishes itself from other methods in two ways. First, it uses the edges of the Delaunay complex and therefore does not require volume calculations of the Voronoi cells. Second, only low-resolution X-ray structures are chosen to represent poorly-packed molecules. This

is in contrast to RosettaHoles2, where predicted structures generated by Rosetta [11] are also included. There exist many scoring methods that separates predicted structures from native structures, but poor packing is one of the things that often distinguishes low-resolution structures from high-resolution ones. The main conclusion of this paper is that the edges of the Delaunay complex characterize packing as well as the volume integration of the Voronoi cell used in RosettaHoles2, but can be computed faster.

II. Methods

The output of the method described here is a *packing cost* which is a quantification of the packing quality of a protein structure. The packing cost of a structure is the average *atom packing cost* of the individual atoms in the structure. The following section describes how the atom packing cost is calculated and why averaging atom packing costs to get the packing cost is reasonable. Finally, the data sets used for training and testing are described.

First, the Delaunay complex of the centers of all heavy atoms, A, is found using the insertion algorithm described by Ledoux [12]. Although this algorithm has a $\mathcal{O}(n^2)$ worst-case running time (where $n = |A|$), in practice it runs fast for two reasons. First, the atoms are inserted in the order they appear in the protein chain. When searching for the tetrahedron containing the inserted point, the method walks from an adjacent tetrahedron of the previously inserted point and, in practice, only traverses a constant number of tetrahedra. Second, the method uses flipping to reinstate the Delaunay criterion after a point is inserted. Since the flipping only affects tetrahedra whose circumsphere contains the newly inserted point, insertion is, in practice, a constant-time operation for evenly distributed points. Assuming that both the point-location and reinstating the Delaunay criterion are expected $\mathcal{O}(1)$ time operations, the algorithm runs in expected $\mathcal{O}(n)$ time.

We define an atom to be *buried* if none of its adjacent tetrahedra are exposed. A tetrahedron, τ, is *exposed* iff there exist a sequence of adjacent tetrahedra, all with circumradii larger than 2.4Å, starting at τ and ending at a tetrahedron which has a face on the convex hull. The radius of 2.4Å is often used as the combined radii of an average heavy atom and a water molecule. Therefore, if a tetrahedron is exposed it indicates that a water molecule can gain access to its interior.

An atom packing cost is assigned to each heavy atom using a feed-forward neural network with 10 input neurons, 20 hidden neurons and 1 output neuron. The values assigned to the input neurons are based on the lengths of edges incident to the atom in the Delaunay complex. Ten bins are defined as shown in Table I. The value of an input neuron is the number of incident edges whose length fall within that bin. When training, the desired atom packing cost for the neural network is 0 if the atom is in a structure with resolution less than 1.8Å and 1 otherwise. The actual output of the neural network is the atom packing cost. Because different types of atoms (carbon, nitrogen, oxygen and sulfur) might appear in different contexts within a protein, a separate neural network is trained for each of the four types of atoms. Sulfur, for instance, has a significantly larger radius than either of the other three atom types. Edges adjacent to a sulfur atom will therefore typically be longer, which is not necessarily an indication of bad packing.

Bin	Interval
0	$[0, 1.15)$
1	$[1.15, 2.04)$
2	$[2.04, 2.13)$
3	$[2.13, 2.44)$
4	$[2.44, 2.72)$
5	$[2.72, 3.01)$
6	$[3.01, 3.34)$
7	$[3.34, 3.71)$
8	$[3.71, 4.13)$
9	$[4.13, \infty)$

Table I
THE INTERVALS OF BINS USED FOR THE 10 INPUTS IN THE NEURAL NETWORK.

The intervals of the bins in Table I are calculated such that any edge incident to a buried atom in the training set has an equal probability of being in any of the bins. To determine these intervals, all edges incident to buried atoms in the training set (defined briefly) are collected in a list. This list is sorted according to the lengths of the edges, and split in ten lists of equal sizes. The last elements of the 9 first lists are used as the boundaries of the bin-intervals.

When training the neural networks to output the atom packing cost, only buried atoms are used as training examples. The reason is that the network might be trained to recognize the size of the protein instead of the packing quality. Non-buried atoms have long adjacent edges, and can be recognized by the number of edges in bin 9. If a neural network is accidentally trained to recognize

the number of non-buried atoms it will have an estimate of the surface area and hence the size of the protein. This is a problem because the average size of low-resolution structures is larger than for high-resolution.

When evaluating the packing cost of a structure, the average atom packing costs of *all* atoms is returned. Averaging over buried atoms only does not significantly affect the packing cost, and since it takes longer to determine which atoms are buried than to calculate the atom packing cost of all atoms, the latter is chosen. Averaging atom costs is justified by inspecting the distribution of atom packing costs. For most proteins this distribution roughly follows a normal distribution which is defined by an average and a standard deviation.

A *training set*, consisting of 3982 protein structures, is retrieved from the PISCES server [13] (pre-compiled data set id: cullpdb_pc40_res3.0_R1.0_d110218). No two structures within this set have sequence similarity higher than 40%. Half of the structures in the training set, the high-resolution structures, have a resolution less than 1.61Å. The other half, the low-resolution structures, have a resolution greater than 2.24Å. All chains that are not specified by the PISCES server are disregarded even though they appear in PDB-files necessary for the test and training sets. Ligands and other heterogeneous atoms (HETATM records) are included and atoms with multiple occupancies are filtered such that only the atom with highest occupancy is included. Only chains with 50 amino acids or more are included. The training set is the basis for all the choices made in the packing cost method and it is used to train the four neural networks.

A *test set*, consisting of 1838 protein structures, is retrieved from the PISCES server such that no two structures in the training set and the test set have more than 40% sequence similarity. As in the training set, half of the structures are high-resolution and the other half are low-resolution. The PDB-files are treated in the same way those in the training set. The test set is used to determine if the packing cost can successfully discriminate between high- and low-resolution structures and is also the basis for the timing experiments in the Results section.

The *CASP9 set*, consisting of all 49899 protein structures submitted to the CASP9 experiment, is retrieved from predictioncenter.org. These structures are examples of computationally generated structures similar to those used as examples of bad packing in RosettaHoles2. This data set is used to confirm the hypothesis that computational structures are poorly-packed and to compare the packing cost to the RosettaHoles2 cost.

III. Results

The experiments seek to illustrate that the packing cost discriminates between well-packed and poorly-packed structures as well as RosettaHoles2, but does it faster.

The discriminatory power of the packing cost is illustrated using distributions of packing costs. Figure 1 shows distributions of packing costs for high- and low-resolution structures in the test set and for structures in the CASP9 set. Figure 2 shows similar distributions for the RosettaHoles2 cost.

The neural networks that determine the packing cost are trained to distinguish high-resolution structures from low-resolution structures so it may seem surprising that the corresponding distributions in Figure 1 are not completely separated. The differences between high- and low-resolution structures can be very subtle so sometimes the packing cost will mis-categorize. As expected, however, most high-resolution structures have a lower packing cost than low-resolution structures and the degree of misclassification is not worse than that of the RosettaHoles2 cost, shown in Figure 2.

Both the packing cost and RosettaHoles2 cost can separate high-resolution structures from CASP9 structures with a high accuracy. This is noteworthy because, unlike the RosettaHoles2 cost, the packing cost is not trained specifically to classify computationally generated structures.

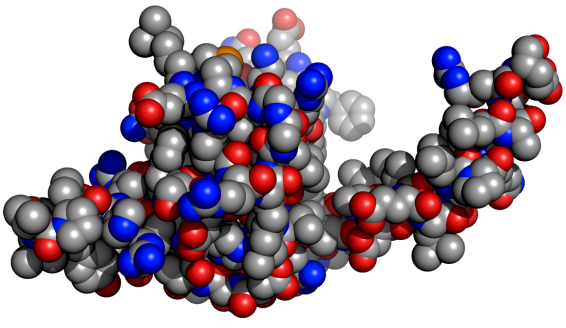


Figure 4. Typical example of a structure with very high RosettaHoles2 cost.

The packing cost and RosettaHoles2 cost both separate high-resolution structures from computer-generated ones, but they may characterize different

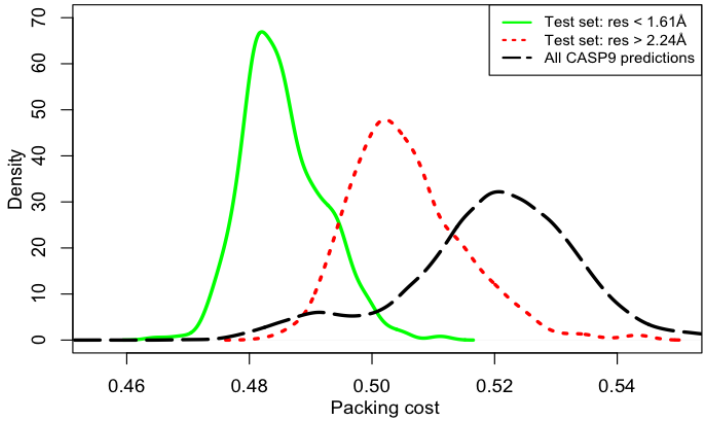


Figure 1. Distributions of packing costs for proteins in the test set and the CASP9 set.

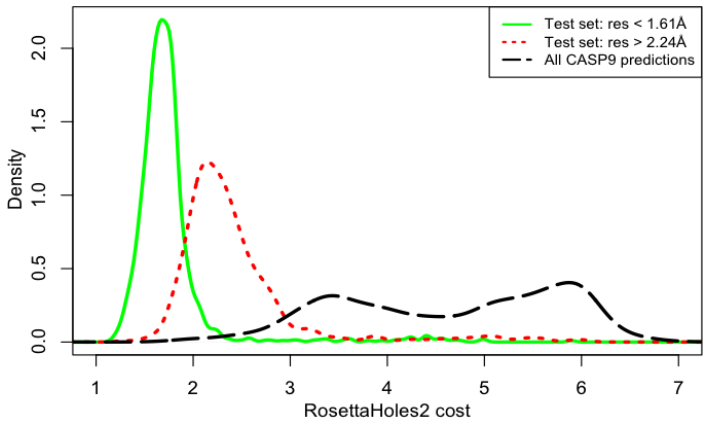


Figure 2. Distributions of RosettaHoles2 costs for proteins in the test set and the CASP9 set.

properties. Figure 3 shows a scatter-plot of RosettaHoles2 costs plotted against packing costs. The main cluster of structures has RosettaHoles2 costs between 1 and 3. Within this cluster there is a clear linear correspondence between the packing cost and the RosettaHoles2 cost (Pearson's squared r of 0.65). There are roughly 100 structures with a RosettaHoles2 cost of more than 3.0. The majority of these are non-globular chains, often with an extended and exposed piece as shown in Figure 4. It is not clear if such structures should be considered well-packed since they are not complete, so it is chosen to disregard these. There are also 15 structures with RosettaHoles2 costs less than 1. It seems that ligands or residues marked as 'unknown' are responsible for most of these, since removing them causes the RosettaHoles2 cost to increase above 1. These are disregarded as well. It is noted that the packing cost is very robust and never returns very extreme values. It is also observed that for the majority of proteins, there is a correlation between the packing cost and the RosettaHoles2 cost.

To demonstrate the improved speed of our method, the system time of the packing cost calculation is measured and displayed as a function of the number of atoms in each structure (Figure 5). The same is done for RosettaHoles2. Both programs are run on a MacBook 2GHz computer and the timing is performed in the source code with `getrusage`. Only the system time of the scoring itself, and not, for example, the time to read the PDB-file, is measured.

For the smallest proteins with less than 500 atoms, the packing cost is calculated between 3 and 4 times faster than the RosettaHoles2 cost. For the larger proteins with roughly 6000 atoms, the packing cost is calculated more than 5 times faster. The computation that dominates out method is finding the Delaunay complex. As mentioned in the Methods section the insertion algorithm uses the chain-structure of the protein to generate

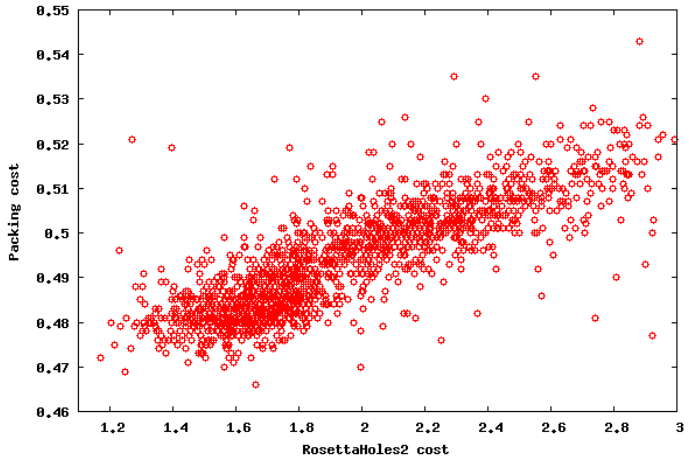


Figure 3. Correlation between packing cost and RosettaHoles2 cost for proteins in the test set.

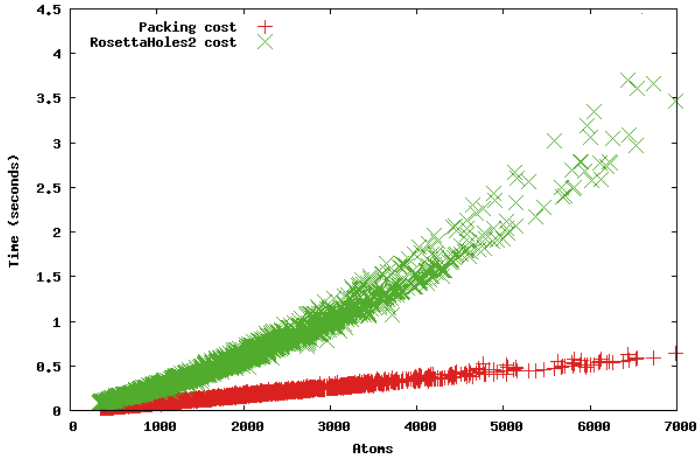


Figure 5. Timing of the packing cost and the RosettaHoles2 cost for proteins in the test set.

the Delaunay complex in expected linear time. This fact is clearly reflected in the timing plot on Figure 5. RosettaHoles2 uses the DAlphaBall program [14], [15] to get the volumes of Voronoi cells. As an intermediate step DAlphaBall finds the Delaunay complex using an insertion and flipping algorithm similar to ours, but it contains a data structure for point-location which gives an expected running time of $\mathcal{O}(n \lg n)$ and does not utilize the chain-structure of proteins.

The ultimate goal of having a fast characterization of the packing cost is to include it as a term in an energy function and improve the packing quality of a protein structure computationally. For a typical protein of $\approx$ 2000 atoms, the packing cost is calculated in $\approx$ 200ms which, in theory, is fast enough to do structure refinement on a massively parallelized system. Furthermore there are a number of ways to improve the speed of the packing cost. Lui and Snoeyink [16], e.g., reports a running time of the tess3 triangulation program that is at least 3 times faster than our insertion algorithm. Guibas and Russel [17] describes how updating the Delaunay complex, after a subset of the points have moved, can be performed faster than recalculating the entire Delaunay complex.

A problem with the packing cost is that many energy functions (Rosetta's, for instance) require their energy terms to be differentiable in order to do fast updates of the energy. In its current form the packing cost is not differentiable. One of the main findings of this paper, however, is that edge-lengths in the Delaunay complex characterize packing just as well as the volume of the Voronoi cells. Since the edge-lengths can easily be differentiated with respect to vertex-

coordinates one can create a differentiable packing cost measure by using a differentiable machine learning method such as support vector machines on distributions of edge-lengths.

IV. Conclusion

An estimate of the packing quality is useful for computational refinement of protein structures. A packing cost was developed and shown to characterize the packing quality of proteins. It was concluded that using edges of the Delaunay complex for characterizing packing is just as efficient as using the Voronoi cells. The observed improvements in speed over previous methods makes it well suited for integration into an energy function.

Acknowledgements

We thank William Sheffler for his kind help in making RosettaHoles2 run properly.

References

[1] R. W. W. Hooft, G. Vriend, C. Sander, and E. E. Abola, "Errors in protein structures," *Nature*, vol. 381, no. 6580, p. 272, 1996.

[2] R. A. Laskowski, M. W. MacArthur, D. S. Moss, and J. M. Thornton, "PROCHECK: a program to check the stereochemical quality of protein structures," *Journal of Applied Crystallography*, vol. 26, no. 2, pp. 283–291, 1993.

[3] M. J. Sippl, "Recognition of errors in three-dimensional structures of proteins," *Proteins*, vol. 17, no. 4, pp. 355–362, 1993.

[4] N. Pattabiraman, K. B. Ward, and P. J. Fleming, "Occluded molecular surface: analysis of protein packing." *Journal of Molecular Recognition*, vol. 8, no. 6, pp. 334–344, 1995.

[5] J. M. Word, S. C. Lovell, T. H. LaBean, H. C. Taylor, M. E. Zalis, B. K. Presley, J. S. Richardson, and D. C. Richardson, "Visualizing and quantifying molecular goodness-of-fit: small-probe contact dots with explicit hydrogen atoms." *Journal of Molecular Biology*, vol. 285, no. 4, pp. 1711–1733, 1999.

[6] M. Gerstein, J. Tsai, and M. Levitt, "The volume of atoms on the protein surface: calculated from simulation, using Voronoi polyhedra." *Journal of Molecular Biology*, vol. 249, no. 5, pp. 955–966, 1995.

[7] A. Poupon, "Voronoi and Voronoi-related tessellations in studies of protein structure and interaction." *Current Opinion in Structural Biology*, vol. 14, no. 2, pp. 233–241, 2004.

[8] K. Rother, P. W. Hildebrand, A. Goede, B. Gruening, and R. Preissner, "Voronoia: analyzing packing in protein structures," *Nucleic Acids Research*, vol. 37, no. suppl 1, pp. D393–D395, 2009.

[9] W. Sheffler and D. Baker, "RosettaHoles: Rapid assessment of protein core packing for structure prediction, refinement, design, and validation," *Protein Science*, vol. 18, no. 1, pp. 229–239, 2008.

[10] ——, "Rosettaholes2: A volumetric packing measure for protein structure refinement and validation," *Protein Science*, vol. 19, no. 10, pp. 1991–1995, 2010.

[11] C. A. Rohl, C. E. M. Strauss, K. Misura, and D. Baker, "Protein structure prediction using rosetta," in *Numerical Computer Methods, Part D*, ser. Methods in Enzymology, L. Brand and M. L. Johnson, Eds. Academic Press, 2004, vol. 383, pp. 66–93.

[12] H. Ledoux, "Computing the 3d Voronoi diagram robustly: An easy explanation," in *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 117–129.

[13] G. Wang and R. L. Dunbrack, "PISCES: a protein sequence culling server," *Bioinformatics*, vol. 19, no. 12, pp. 1589–1591, 2003.

[14] H. Edelsbrunner and P. Koehl, "The weighted-volume derivative of a space-filling diagram," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2203–2208, 2003.

[15] R. Bryant, H. Edelsbrunner, P. Koehl, and M. Levitt, "The area derivative of a space-filling diagram," *Discrete & Computational Geometry*, vol. 32, pp. 293–308, 2004.

[16] L. Yuanxin and J. Snoeyink, *Combinatorial and Computational Geometry*. New York, NY, USA: Cambridge University Press, 2005, ch. 23, pp. 439–458.

[17] L. Guibas and D. Russel, "An empirical comparison of techniques for updating delaunay triangulations," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ser. SCG '04. New York, NY, USA: ACM, 2004, pp. 170–179.

5.8 Visualizing and Representing the Evolution of Topological Features

The following 2 pages contains the extended abstract "Visualizing and representing the evolution of topological features. R. Fonseca and D. M. S. Jørgensen. Extended abstract for CG:YRF at SoCG. 2012" [64].

Visualizing and representing the evolution of topological features

Rasmus Fonseca* Desirée Malene Schreyer Jørgensen†

Abstract

Simplicial complexes are discrete representations of topological spaces that are practical for computational studies. The first three Betti-numbers (indicating the number of components, tunnels and voids), as well as the topological persistence of each such feature, is well-defined and can be efficiently computed for simplicial complexes embedded in 2D and 3D [1, 2].

We introduce a novel representation of the evolution of topological features in simplicial complexes using so-called tunnel-trees in 2D and void-trees in 3D. This new representation makes it possible to analyze topological evolution by applying tools for analysis of binary trees. Furthermore it supplies a new method for visualizing topological evolution.

Introduction

A simplicial complex, $\mathcal{K}$, is a set of simplices where any face of a simplex in $\mathcal{K}$ is also in $\mathcal{K}$ and the intersection of two simplices in $\mathcal{K}$ is either empty or a face of both simplices. Delfinado and Edelsbrunner [1] define a *filter* to be a sequence of simplices, $\sigma^1, \sigma^2, \ldots, \sigma^n$, where $\mathcal{K}_i = \{\sigma^1, \sigma^2, \ldots, \sigma^i\}$ is a simplicial complex for any choice of i (see left part of Figure 1). The filter represents the evolution of a simplicial complex and will be the focus of the methods described here. The topological features of a complex can be described using the Betti-numbers, β_d, which indicate the rank of the dth homology group. The first three Betti-numbers $(\beta_0, \beta_1, \beta_2)$ can be interpreted more intuitively as the number of components, holes, and voids respectively. A $\mathcal{O}(n\alpha(n))$-time algorithm exists to calculate the evolution of β_d as a simplicial complex is grown using a filter [1]. This method identifies each k-simplex, σ^i, as either *positive* if it creates a new k-cycle and thereby increases β_k, or *negative* if it changes a k-cycle into a k-boundary and thereby decreases β_{k-1}. For each positive k-simplex, σ^i, the negative $(k+1)$-simplex, σ^j, that is responsible for turning the k-cycle, created by σ^i, into a k-boundary can be efficiently identified [2]. The difference between the indices of such two simplices is defined to be the *persistence* of the k-cycle represented by σ^i.

*Department of Computer Science, University of Copenhagen, rfonseca@diku.dk

†Department of Computer Science, University of Copenhagen, daisy@diku.dk

Tunnel- and void-trees

One interesting observation about tunnels in simplicial complexes embedded in 2D is that, often, when a positive 1-simplex (edge) is added to the complex, it splits one tunnel in two. If the empty space around the complex is considered a *bounding tunnel*, then every positive edge will split an existing tunnel in two. Similarly, if the entire space around a simplicial complex embedded in 3D is considered a *bounding void*, then a positive 2-simplex (triangle) always splits an existing void in two.

Based on this observation we define a *tunnel-tree* (or β_1-tree) of a 2D filter to be a binary tree where each node represents a distinct tunnel (see right part of Figure 1). The root is the bounding tunnel, and the leaves are triangular tunnels that will not be split further. With each node n we associate the positive edge that represents the tunnel, $\epsilon(n)$, and with each leaf, we associate the negative triangle that fills this tunnel, $\tau(n)$. The tunnel-tree is ordered such that for any node n, the triangle of the rightmost leaf, $\tau(\textsc{Tree-Max}(n))$, is the triangle that 'destroys' $\epsilon(n)$ and hence determines its persistence. A *void-tree* (or β_2-tree) of a 3D filter is defined in a similar fashion, only with positive triangles as nodes and negative tetrahedra as leaves.

A β_k-tree is constructed by running through the filter backwards as shown in Algorithm 1. Leaves are created when a negative $(k+1)$-simplex is encountered and the roots of leaves are connected when positive k-simplices are encountered.

Algorithm 1 Build a β_k-tree given a filter

Create a 'bounding node', n_b
for $i = n$ **to** 1 **do**
  if σ^i is a negative $(k+1)$-simplex **then**
    Create a new node, n, and set $\tau(n) \leftarrow \sigma^i$
  else if σ^i is a positive k-simplex **then**
    $(n_0, n_1) \leftarrow$ Nodes of the two $(k+1)$-simplices adjacent to σ^i
    $(n_0, n_1) \leftarrow (\textsc{Root}(n_0), \textsc{Root}(n_1))$
    Swap n_0 and n_1 if $\tau(\textsc{Tree-Max}(n_0))$ is younger than $\tau(\textsc{Tree-Max}(n_1))$
    Create a new node n with $n.left \leftarrow n_0$, $n.right \leftarrow n_1$, and $\epsilon(n.left) \leftarrow \sigma^i$
  end if
end for
return $\textsc{Root}(n_b)$

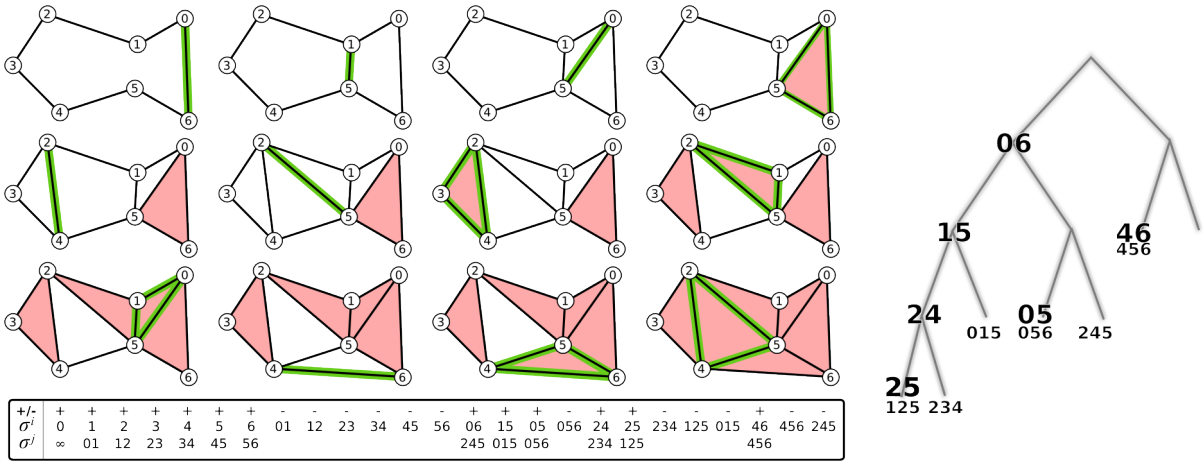


Figure 1: **Left:** A 2D filter. For all positive k-simplices, σ^i, the $(k+1)$-simplex, σ^j, responsible for turning the k-cycle, represented by σ^i, into a k-boundary is indicated as well. **Right:** The tunnel-tree (β_1-tree) of the filter. Both $\epsilon(n)$ and $\tau(n)$ are shown for each node if they are defined.

In line 4, the $(k+1)$-simplex can be associated with its node using a hash-map. This ensures that locating the nodes of adjacent $(k+1)$-simplices in line 6 can be performed in constant time. In line 6, if one of the $(k+1)$-simplices adjacent to σ^i is not defined then the bounding node n_b is used instead. If σ^i has no adjacent $(k+1)$-simplices then a new node is created for n_0, and n_1 is set to n_b. Line 8 guarantees that the youngest simplex in a subtree can always be found by going to the far right in the tree using TREE-MAX.

A β_k-tree may be arbitrarily unbalanced, so a straightforward implementation will run in $\mathcal{O}(n^2)$ time worst case. The TREE-MAX-method can be improved to $\mathcal{O}(1)$ time by maintaining the maximum of each subtree as they are constructed. A data structure similar to disjoint-sets can be used to make the ROOT method run in $\mathcal{O}(\alpha(n))$-time, so the entire method runs in $\mathcal{O}(n\alpha(n))$ worst case time.

Applications

One attractive property of β_k-trees is that they give an alternative representation of the topological evolution of a filter. This can be used in several ways.

First, the fact that simplices in the subtree of a particular node will tend to be spatially close to each other gives rise to a new definition of *local persistence*. A particular edge, representing a tunnel, might be deemed particularly persistent if its subtree contains more than a certain number of nodes. Such a definition of persistence will not be affected by the addition of simplices outside the tunnel.

Using a Delaunay complex and the radius of the smallest empty circumcircle to generate an α-filter [3], the arrangement of a particular sub-tree also gives an indication of the shape of the corresponding feature. For instance, a node with an unbalanced sub-tree indicates a tunnel that is narrowing, whereas a balanced node indicates a constant width.

For some applications, a tree might be a better visualization of the topological evolution than e.g. k-triangles [2]. The above mentioned properties of locality can be computationally analyzed, but they can also be derived simply by inspecting β_k-trees. The length of edges in the tree can furthermore be scaled to reflect the difference in birth time of the $\epsilon(n)$ simplices.

Another interesting property of β_k-trees is that all $(k+1)$-simplices within a particular tunnel/void are easily identified by locating the node in the tree with the desired $\epsilon(n)$ and then collecting all leaves in the subtree using any tree-traversal method. In this manner the area of tunnels/volume of voids, for instance, is easily calculated.

Finally, any analysis method that works on trees is now applicable to topological evolutions. For instance the topology of two point-sets can be compared by finding the tree-edit-distance between the tunnel-trees (or void-trees) of their respective α-filters.

References

[1] C. Delfinado and H. Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995.

[2] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[3] H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. In *Proceedings of the 1992 Workshop on Volume Visualization*, pages 75–82. ACM, 1992.

Bibliography

[1] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.3r1. August 2010.

[2] Carl-Ivar Branden and John Tooze. *Introduction to Protein Structure*. Garland Publishing, second edition, 1999.

[3] Erion Plaku, Hernn Stamati, Cecilia Clementi, and Lydia E. Kavraki. Fast and reliable analysis of molecular motion using proximity relations and dimensionality reduction. *Proteins: Structure, Function, and Bioinformatics*, 67(4):897–907, 2007.

[4] Computational biology - fast and reliable analysis of molecular motion (dpes-scimap). `http://faculty.cua.edu/plaku/ResearchCompBio.html`. Online; accessed: 21/07/2012.

[5] Glennie Helles. A comparative study of the reported performance of ab initio protein structure prediction algorithms. *Journal of the Royal Society Interface*, 5(21):387–396, 2008.

[6] David Lipman, Alexander Souvorov, Eugene Koonin, Anna Panchenko, and Tatiana Tatusova. The relationship of protein conservation and sequence length. *BMC Evolutionary Biology*, 2(1):20, 2002.

[7] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[8] John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, and Anna Tramontano. Critical assessment of methods of protein structure prediction (casp)round ix. *Proteins: Structure, Function, and Bioinformatics*, 79(S10):1–5, 2011.

[9] Wikipedia. File:ramachandran plot general 100k, 2010. [Online; accessed 21/07/2012].

[10] Walter Pirovano and Jaap Heringa. Protein secondary structure prediction data mining techniques for the life sciences. In Oliviero Carugo and Frank Eisenhaber, editors, *Data Mining Techniques for the Life Sciences*, volume 609 of *Methods in Molecular Biology*, chapter 19, pages 327–348. Humana Press, Totowa, NJ, 2010.

[11] Sven Griep and Uwe Hobohm. Pdbselect 1992-2009 and pdbfilter-select. *Nucleic Acids Research*, 38(suppl 1):D318–D319, 2010.

[12] Glennie Helles. Improving search for low energy protein structures with an iterative niche genetic algorithm. In *Bioinformatics'10*, pages 226–232, 2010.

[13] Wikipedia. Root-mean-square deviation (bioinformatics) — wikipedia, the free encyclopedia, 2012. [Online; accessed 12-September-2012].

[14] Wouter Boomsma, Kanti V Mardia, Charles C Taylor, Jesper Ferkinghoff-Borg, Anders Krogh, and Thomas Hamelryck. A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(26):89327, Jul 2008.

[15] Rui Kuang, Christina S Leslie, and An-Suei Yang. Protein backbone angle prediction with machine learning approaches. *Bioinformatics (Oxford, England)*, 20(10):161221, Jul 2004.

[16] Olav Zimmermann and Ulrich H E Hansmann. Support vector machines for prediction of dihedral angle regions. *Bioinformatics (Oxford, England)*, 22(24):300915, Dec 2006.

[17] Martin Paluszewski and Pawel Winter. Protein decoy generation using branch and bound with efficient bounding. *Proc. of the 8th Int. Workshop, WABI 2008, LNBI 5251*, pages 382–393, 2008.

[18] Martin Paluszewski and Pawel Winter. Ebba: Efficient branch and bound algorithm for protein decoy generation. *Department of Computer Science, Univ. of Copenhagen*, 8(08), 2008.

[19] Thomas Hamelryck. An amino acid has two sides: A new 2d measure provides a different view of solvent exposure. *Proteins: Structure, Function, and Bioinformatics*, 59(1):38–48, 2005.

[20] Jiangning Song, Hao Tan, Kazuhiro Takemoto, and Tatsuya Akutsu. Hsepred: predict half-sphere exposure from protein sequences. *Bioinformatics*, 24(13):1489–1497, 2008.

[21] K A Dill, S Bromberg, K Yue, K M Fiebig, D P Yee, P D Thomas, and H S Chan. Principles of protein folding–a perspective from simple exact models. *Protein science : a publication of the Protein Society*, 4(4):561602, Apr 1995.

[22] Volker A. Eyrich, Daron M. Standley, Anthony K. Felts, and Richard A. Friesner. Protein tertiary structure prediction using a branch and bound algorithm. *Proteins*, 35(1):41–57, 1999.

[23] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm. Technical report, MEC, Cardiff University, UK, 2005.

[24] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes Univ., Engineering Faculty, Computer Engineering Department, Nov. 2005.

[25] Alexander D. MacKerell, Nilesh Banavali, and Nicolas Foloppe. Development and current status of the CHARMM force field for nucleic acids. *Biopolymers*, 56(4):257–265, 2000.

[26] J. L. Klepeis and C. A. Floudas. ASTRO-FOLD: a combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. *Biophysical journal*, 85(4):2119–2146, October 2003.

[27] Jianlin Cheng and Pierre Baldi. Three-stage prediction of protein β-sheets by neural networks, alignments and graph algorithms. *Bioinformatics*, 21(suppl 1):i75–i84, June 2005.

[28] Jieun Jeong, Piotr Berman, and Teresa M. Przytycka. Improving strand pairing prediction through exploring folding cooperativity. *IEEE/ACM Trans. Comp. Bio. Bioinf.*, 5:484–91, 2008.

[29] M. Lippi and P. Frasconi. Prediction of protein beta-residue contacts by Markov logic networks with grounding-specific weights. *Bioinformatics*, 25:2326–33, 2009.

[30] R. Rajgaria, Y. Wei, and C. A. Floudas. Contact prediction for beta and alpha-beta proteins using integer linear optimization and its impact on the first principles 3d structure prediction method astro-fold. *Proteins*, 78(8):1825–1846, 2010.

[31] Ingo Ruczinski, Charles Kooperberg, Richard Bonneau, and David Baker. Distributions of beta sheets in proteins with application to structure prediction. *Proteins: Structure, Function, and Genetics*, 48(1):85–97, July 2002.

[32] Brent Wathen and Zongchao Jia. A hierarchical order within protein structures underlies large separations between strands in β-sheets. *Proteins: Structure, Function, and Bioinformatics*, 2012.

[33] Nelson Max, ChengCheng Hu, Oliver Kreylos, and Silvia Crivelli. BuildBeta - a system for automatically constructing beta sheets. *Proteins: Structure, Function, and Bioinformatics*, 78(3):559–574, 2010.

[34] Gaurav Porwal, Swapnil Jain, S. Dhilly Babu, Deepak Singh, Hemant Nanavati, and Santosh Noronha. Protein structure prediction aided by geometrical and probabilistic constraints. *Journal of Computational Chemistry*, 28(12):1943–1952, 2007.

[35] J.W. Ponder. Tinker software tools for molecular design, version 4.2, washington university school of medicine, st. louis, usa, 2004.

[36] Wikipedia. Sequential structure alignment program — wikipedia, the free encyclopedia, 2011. [Online; accessed 12-September-2012].

[37] William R. Taylor and Christine A. Orengo. Protein structure alignment. *Journal of Molecular Biology*, 208(1):1 – 22, 1989.

[38] Philip Bradley and David Baker. Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. *Proteins: Structure, Function, and Bioinformatics*, 65(4):922–929, 2006.

[39] Wikipedia. Global distance test — wikipedia, the free encyclopedia, 2012. [Online; accessed 12-September-2012].

[40] Adam Zemla. LGA: A method for finding 3D similarities in protein structures. *Nucleic acids research*, 31(13):3370–3374, July 2003.

[41] Eunhee Koh and Taehyo Kim. Minimal surface as a model of β-sheets. *Proteins: Structure, Function, and Bioinformatics*, 61(3):559–569, 2005.

[42] Eunhee Koh, Taehyo Kim, and Hyun-soo Cho. Mean curvature as a major determinant of β-sheet propensity. *Bioinformatics*, 22(3):297–302, 2006.

[43] Jie Liang, Herbert Edelsbrunner, Ping Fu, Pamidighantam V. Sudhakar, and Shankar Subramaniam. Analytical shape computation of macromolecules: I. molecular area and volume through alpha shape. *Proteins Structure Function and Genetics*, 33(1):1–17, 1998.

[44] Itay Lotan, Fabian Schwarzer, Dan Halperin, and Jean-Claude Latombe. Algorithm and data structures for efficient energy maintenance during monte carlo simulation of proteins. *Journal of Computational Biology*, 11(5):902–932, 2004.

[45] Dan Halperin and Mark H. Overmars. Spheres, molecules, and hidden surface removal. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 113–122. ACM, 1994.

[46] William Sheffler and David Baker. Rosettaholes: Rapid assessment of protein core packing for structure prediction, refinement, design, and validation. *Protein science*, 18(1):229–239, 2009.

[47] William Sheffler and David Baker. Rosettaholes2: A volumetric packing measure for protein structure refinement and validation. *Protein Science*, 19(10):1991–1995, 2010.

[48] Mark de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, Second Edition*. Springer, 2nd edition, 2000.

[49] Cecil J.A. Delfinado and Herbert Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995.

[50] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[51] Anne Micheli and Dominique Rossin. Edit distance between unlabeled ordered trees. *ITA*, pages 593–609, 2006.

[52] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[53] Wikipedia. Hierarchical clustering — wikipedia, the free encyclopedia, 2012. [Online; accessed 15-September-2012].

[54] Vin de Silva and Gunnar Carlsson. Topological estimation using witness complexes. In M. Alexa and S. Rusinkiewicz, editors, *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2004.

[55] Y. Zhang. Progress and challenges in protein structure prediction. *Current opinion in structural biology*, 18(3):342–348, 2008.

[56] Yaoqi Zhou, Yong Duan, Yuedong Yang, Eshel Faraggi, and Hongxing Lei. Trends in template/fragment-free protein structure prediction. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 128:3–16, 2011. 10.1007/s00214-010-0799-2.

[57] Glennie Helles and Rasmus Fonseca. Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks. *BMC Bioinformatics*, 10(1):338, 2009.

[58] Rasmus Fonseca, Martin Paluszewski, and Pawel Winter. Protein structure prediction using bee colony optimization metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 9(2):181–194, 2010.

[59] Rasmus Fonseca, Martin Paluszewski, and Pawel Winter. Protein structure prediction using bee colony optimization metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 9(2):181–194, 2010.

[60] Rasmus Fonseca, Glennie Helles, and Pawel Winter. Ranking beta sheet topologies with applications to protein structure prediction. *Journal of Mathematical Modelling and Algorithms*, 10(4):1–13, 2011.

[61] Pawel Winter and Rasmus Fonseca. Adjustable chain trees for proteins. *Journal of Computational Biology*, 19(1):83–99, 2012.

[62] Rasmus Fonseca and Pawel Winter. Bounding volumes for proteins: A comparative study. Accepted: Journal of Computational Biology, 2012.

[63] Rasmus Fonseca, Pawel Winter, and Kevin Karplus. Protein packing quality using delaunay complexes. In *Eighth International Symposium on Voronoi Diagrams*, pages 117–122. IEEE, 2011.

[64] Rasmus Fonseca and Desiree M. S. Jørgensen. Visualizing and representing the evolution of topological features. In *Symposium of Computational Geometry - Young Researchers Forum*, pages 29–30, 2012.