Searching for the Native Structures of Proteins

PhD thesis

by

Glennie Helles

University of Copenhagen, Department of Computer Science
The PhD School of Science, Faculty of Science, University of Copenhagen, Denmark

May 17, 2010

Supervisor:
Associate Professor Martin Zachariasen

Contents

Preface

This PhD thesis has been prepared at the Department of Computer Science, University of Copenhagen, under academic supervision of Associate Professor Martin Zachariasen.

The thesis includes five papers, four of which have already been published - the fifth is currently under review. Co-authors are clearly listed where appropriate. Most of this thesis, and thus also most of the papers, pertains to protein structure prediction, which is the main topic of this thesis. One paper stands out by relating to a different biological area: population genetics. The paper has been included in the final chapter entitled Other work".

The scientific work and contributions are documented in the papers in this thesis. It should be emphasized that the introductory notes to the chapters are not intended to provide a comprehensive introduction to the research areas. Rather they have been kept short and concise and are intended only as a quick brush up on the topics. The readers of this thesis are thus expected to have some background knowledge of the research area.

Acknowledgments

A number of people have provided priceless feedback and help during my PhD study. I would like to thank Martin Zachariasen, Professor David Pisinger and the PhD students of the former Algorithms and Optimization group who regrettably transferred to DTU last winter. You provided invaluable support and a cheerful atmosphere during a difficult time. I would also like to thank Professor Teresa Head-Gordon at the University of California in Berkeley, with whom I had the pleasure of working with during my stay abroad. Finally, I want to pay special thanks to Associate Professor Pawel Winter, who more than anyone else has made me feel like I belonged in research, and to fellow PhD student Rasmus Fonseca for sharing my interest in protein structure prediction and for being a fantastic colleague.

On a more personal level, I want to thank my parents for their constant love and support, Bo Lange for his immense patience and all of my friends for listening to me as I rambled on about protein structure prediction. Last but not least, I would like to pay a very special thanks to my ex-husband, Morten Helles, without whom I would probably never have had the confidence to start this PhD. Thank you for never letting me doubt that I could do this.

Chapter 1

Introduction

1.1 Motivation

Proteins are essential in all living organisms. They are key components in everything from cell structure and inter-cell signaling to muscle tone and digestion. Because of their widespread functioning, many illnesses are directly linked to proteins. General lack of proteins will cause symptoms like weakening of muscles, anemia and low heart rate. Misfolded (deform) proteins cause a number of very serious illnesses, like CreutzfeldtJakob disease, Alzheimer's and Cystic Fibrosis. Understanding how proteins fold is therefore of key importance not only from a scientific point of view, but from a medical point of view.

All proteins are made up by a sequence of amino acids - referred to as the primary sequence - that folds up into a unique and sequence dependent three-dimensional structure within seconds of their synthesis. This structure unequivocally determines the function of the protein. In 1973, Nobel Prize laureate Christian Anfinsen published his paper on "Principles that Govern the Folding of Protein Chains" [1], which argued convincingly that the structure of a protein is determined solely by the sequence of amino acids, and that proteins adopt the configuration with the lowest free energy. This hypothesis has inspired a number of researchers in both biology, biophysics and computer science to attack this problem in order to "crack" the code and predict the three-dimensional structures of proteins from their primary sequences alone, however so far with limited success. There are two main obstacles to this problem: finding an appropriate and sufficiently accurate way of estimating the energy of a protein, and finding a method that will take us to the lowest energy structures quickly and reliably.

Estimating the energy of a protein can be done to various degrees of precision. Highly accurate calculations are very time consuming though, and less accurate calculations may in fact suffice. To be of any use in protein structure prediction, however, the only thing that must be highly accurate is the mapping between the native structure and the lowest energy. If this mapping is off and the native protein is not the structure with the lowest energy, even the most thorough prediction method is bound to fail.

Devising an algorithm that can efficiently take us to the lowest energy structure reliably seems almost unachievable since, theoretically, there are infinitely many ways a protein can fold. The backbone of each amino acid has two degrees of freedom in a continues search space that grows exponentially with the number of amino acids in the protein. Simply evaluating all possible chain configurations is not possible. Instead, one of two different approaches are employed in practice: either we search for the lowest energy structure or we attempt to simulate to folding process thereby arriving at the lowest energy structure. Simulation is likely to yield the best results, but unfortunately it is much too slow (in the order of hundreds of thousands of CPU years for even the smallest of proteins), and searching is thus the choice of preference by most research groups. Of course, searching can be done in many different ways – as will be described in much further details in chapter 2 – and there has yet to be defined a consensus in the community of the most efficient search strategy for this particular problem.

In 1994, the first CASP (Critical Assessment of Techniques for Protein Structure Prediction) competition was held and has since been held every other year. It is the ultimate blind test where research groups can benchmark their prediction algorithms against one another on proteins of yet unknown structure. The first couple of rounds of CASP did not yield any remarkable results, but then prediction accuracy was suddenly improved significantly for certain proteins, namely the ones for which the Protein Data Bank (PDB) contained other proteins with high sequence identity to the unknown protein (so-called homologs). Since then, proteins entered in CASP are split into two categories: template modeling (proteins that contain a homolog in PDB) and *ab initio* prediction (proteins that do not contain a homolog in PDB).

In this thesis, I focus solely on *ab initio* folding and specifically on ways that can efficiently improve the search for low energy structures.

1.2 Goals

The focus of this thesis is *ab initio* protein structure prediction and the main goal is to contrive, analyze and experiment with methods and theories that can facilitate faster and more reliable predictions. After a thorough survey of previously published *ab initio* PSP methods (Chapter 3), which provided good insight into formerly explored paths, I have attempted to reach this goal by attacking the problem from two different but complementary angles. On one hand, I have explored how parallelism can be used to greatly enhance search efficiency of meta-heuristics, and on the other hand, I have looked at different ways to decrease the conformational search space to make it more tractable.

1.3 Achievements and Contributions

I present five papers in this thesis. Four of them relate directly to protein structure prediction while the fifth paper concerns an experiment with the application of genetic algorithms within another biological area: population genetics.

The first paper (presented in chapter 2) provides an in-depth survey and comparison of existing methods for *ab initio* PSP mehods. Such a survey has not previously been published. It is especially useful for the computer scientists of the PSP research community who are interested in algorithmic aspects of PSP and need a good overview of current trends and results. Generally, little attention is paid to the differences in algorithmic performance in this research field, and standardized test sets are completely absent, which makes it difficult for computer scientists to investigate the exact effect of the choice of algorithm. In this survey, I attempt to overcome this problem by identifying all parameters that may influence performance and then compare the settings of these parameters for all recently published *ab initio* PSP methods.

The second paper, presented in chapter 3 (short version in Appendix A), addresses algorithmic performance in PSP. Parallel meta-heuristics are generally known to outperform sequential meta-heuristics, but parallel versions appear to have been explored to a rather limited extend in the field of PSP. In this report, however, we show experimentally the vast improvements that can be achieved with parallel meta-heuristics in PSP if parallelism is thought into the design of the algorithms, rather than just being seen as a way to run multiple simulations at the same time (which to a large extend is the trend of current PSP algorithms). Specifically, we propose a parallel iterative niche genetic algorithm (inGA), and show how it outperforms parallel runs of sequential algorithms, simulated annealing and a genetic algorithm, as well as the two parallel algorithms: the parallel tempering algorithm and the traditional niche genetic algorithm.

The third paper (chapter 4) presents joint work with Rasmus Fonseca. We use a neural network to predict a probability distribution for coil residues in proteins which can help guide search algorithms to the most probable dihedral angle space of coil residues. While the dihedral angles of helical and strand residues are usually confined to a rather small area of the Ramachandran plot, coil residues are much more random. Predicting the structure of coil residues is thus a most challenging issue in *ab initio* PSP.

The forth paper (chapter 4) presents joint work with Rasmus Fonseca and Pawel Winter. We look at the currently very challenging job of predicting β-topologies for proteins that would be of significant importance in PSP if it could be done accurately. In particular, we investigate the quality of two β-topology scoring functions and suggest an enumeration scheme that identifies and ranks β-topologies. Ranking β-topologies allows us to make realistic estimations of how many β-topologies we need to sample for a given protein in order to make it very likely that we have included the native β-topology.

The fifth paper is presented in chapter 5. It is not related to PSP but to another biological research field: population genetics. Interestingly enough, despite the source of inspiration, genetic algorithms have apparently never previously been explored as a simulation tool for population geneticists. In this paper, we present a genetic algorithm that simulates evolution of so-called *ebony* mutants in a population of Drosophila *melanogasters*, also known as common fruit flies. Results from the simulation are directly compared to real life experiments and show close to identical development. Genetic algorithms thus seem quite adequate at mimicking the dynamics of evolution and seem highly suitable as a simulation tool in population genetics.

1.4 Outline

The thesis contains four main chapters: 2, 3, 4 and 5. Each of the chapters contains a short introduction to the research field leading up to the papers. Chapter five includes two papers, and the other three chapters include one paper each.

- **Chapter 2** is an introduction to the *ab initio* PSP research field. It provides short background information about proteins and presents the protein folding problem. The chapter is concluded by the paper "A Comparative Study of the Reported Performance of *Ab Initio* Protein Structure Prediction Algorithms" which provides an in-depth survey and comparison of existing methods for *ab initio* PSP.
- **Chapter 3** opens with a brief introduction to meta-heuristics in protein structure prediction. This leads up to the paper "Exploring Parallel Metaheuristics for Protein Structure Prediction", a technical report that explores and compares the efficiency of parallel meta-heuristics in PSP. A short published version of the technical report is included in Appendix A.
- **Chapter 4** deals with ways to restrict the conformational search space of the PSP problem. Aside from a brief introduction, it includes two papers "Predicting Dihedral Angle Probability Distributions for Protein Coil Residues from Primary Wequence using Neural Networks" and "Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction" that each offers a specific way of limiting the search space.
- **Chapter 5** bears the title "Other work" and includes the final paper "Simulating Evolution of *Drosophila melanogaster Ebony* Mutants Using a Genetic Algorithm" on the applicability of genetic algorithms in the population genetics research field.

The conclusions are summed up in chapter 6.

Chapter 2

Ab initio Protein Structure Prediction

2.1 Background

Currently, the three-dimensional (tertiary) structure of a protein can only be determined reliably by time consuming experiments, and the tertiary structure has thus only been determined for a fraction of all the identified protein sequences. For many years, an ongoing effort has been made to devise a computer algorithm that could predict the tertiary structure from the primary sequence, but it has proven to be a very challenging task indeed. In *ab initio* protein structure prediction, one attempts to come up with a method or algorithm that can predict the structure of a protein without any knowledge of globally similar proteins. This should theoretically be possible, because when a protein folds into its three-dimensional structure it essentially follows the basic laws of physics – and it always folds into the same structure. Different amino acids have different chemical properties, and atoms are thus constantly attracted to and repelled by each other, drawing some amino acids close in proximity and pushing others far apart. Unfortunately, the many degrees of freedom encountered in proteins raise the complexity of the problem to a level where algorithms cannot seem to effectively utilize this information to predict the structure, and approximations always seem to be too crude to be reliable.

2.2 The basic structure

A protein is a macromolecule made up by a number of smaller molecules known as amino acids. 20 different amino acids – or *residues* – are encountered in proteins (listed in Figure 2.3). Every amino acid consists of a backbone and a side chain. The backbone is shown in Figure 2.1, and it is identical for all amino acids. The side chains, on the other hand, differ from each other both in the number of atoms, physical structure and chemical characteristics. In proteins, the backbone of one amino acid forms a covalent bond with the backbones

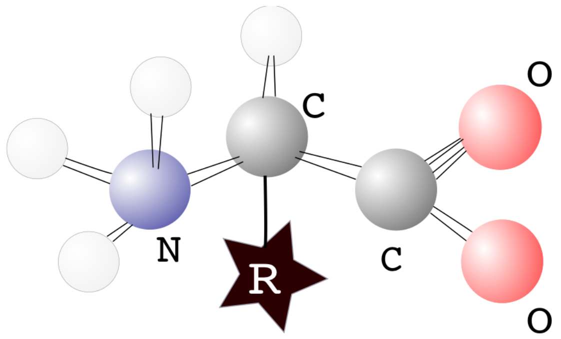


Figure 2.1: Common backbone structure of the 20 amino acids encountered in proteins

of the previous and succeeding amino acids - unless it is first or last residue, in which case it is only bonded to the succeeding or previous amino acid, respectively.

The covalent bond is called a peptide bond, and unlike the bonds between the other backbone atoms, this bond is partially double bonded and thus very rigid. In practice, this means that the atoms around the N-C bond are more or less locked in a plane, leaving only atoms around the C-C_α bond and the C_α-N bond (see Figure 2.2) to rotate (relatively) freely. The dihedral angle (rotation) around the N-C bond is referred to as the ω-angle, and it is close to 180° for nearly all peptide bonds. The only amino acid that frequently deviates from this *trans*-conformation is proline, which due to its circular structure almost equally often has an ω-angle of close to 0° (*cis*-conformation). The dihedral angle around the C-C_α bond is referred to as the ϕ-angle and the dihedral angle around the C_α-N bond is referred to as the ψ-angle. Theoretically, atoms can rotate completely freely around the two latter bonds although, as discussed elsewhere in this thesis, in practice they do not. The flexibility does, however, still mean that the number of possible ways a protein can fold grows exponentially with the number of amino acids in the chain.

The three-dimensional structure of a protein is adopted due to a mixture of biochemical interactions. Probably the most significant contributer – often referred to as the driving force of protein folding – is the hydrophobic effect. Some amino acids are highly hydrophobic,

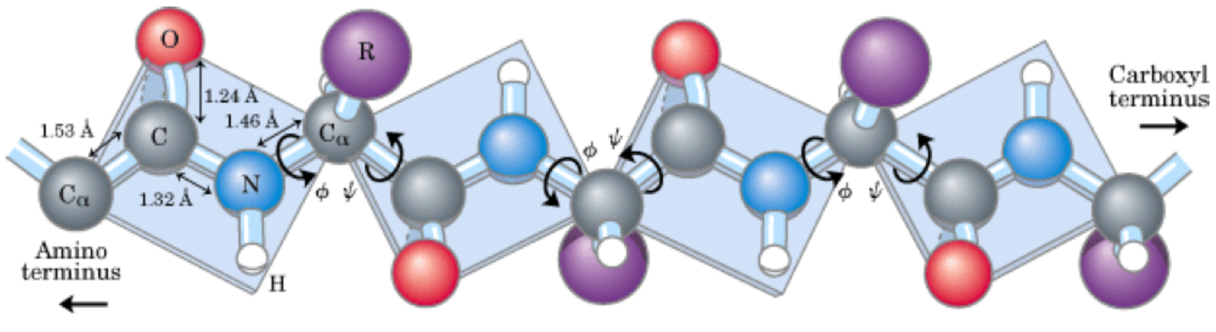


Figure 2.2: The planar structure of peptide bonds

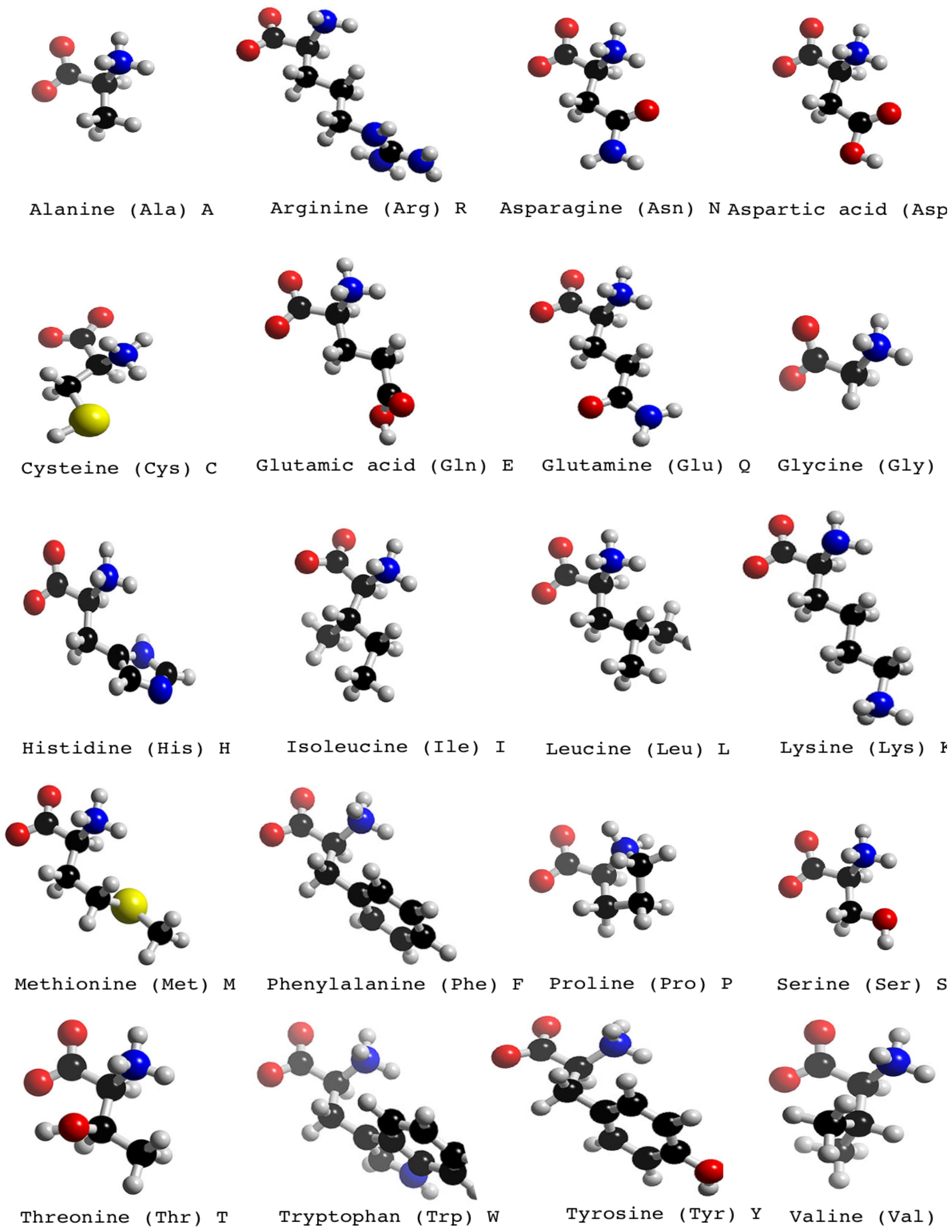


Figure 2.3: The 20 amino acids

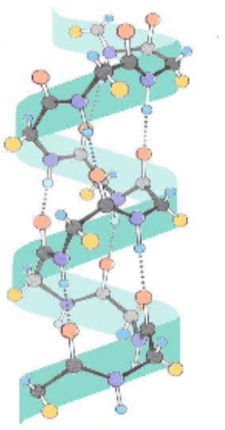


Figure 2.4: Illustration of an α-helix

and they quickly seek to the core of the protein far away from the watery surface. The hydrophilic amino acids are thus pushed to the surface of the protein, forming a characteristic hydrophobic core and hydrophilic surface. While the hydrophobic effect is probably largely responsible for the overall structure, two other contributers, the van der Waals force and hydrogen bonds, play significant roles in creation and maintenance of local structures. The van der Waals force is the attractive/repulsive force between atoms caused by the fluctuating polarization of molecules that arises because one side of a molecule is always somewhat positive and the opposite side somewhat negative. A hydrogen bond, on the other hand, typically forms between a hydrogen atom which is attached to a electronegative atom - like nitrogen atoms in proteins - and another electronegative atom like oxygen. The bond is not as strong as a covalent bond, but it is stronger than the van der Waals force, and it plays a vital role in the different local structures encountered in proteins.

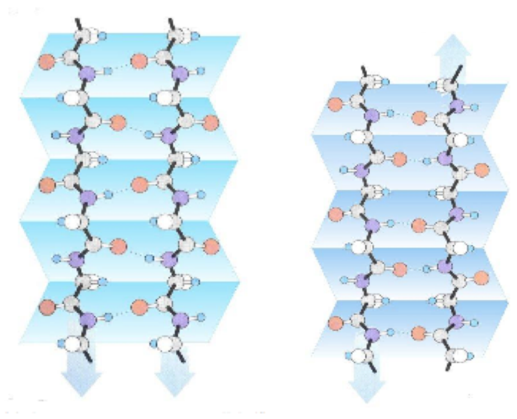


Figure 2.5: Illustration of parallel (left) and antiparallel (right) arrangements of β-strands

2.2.1 Local structures

The dihedral angles adopted by the amino acids are influenced by both long-range and short-range interactions. However, steric clashes between atoms in flanking residues put significant restraints on the dihedral angles actually observed in proteins, regardless of their primary sequence. This can be illustrated with a Ramachandran plot, like the one in Figure 2.6, that shows ϕ-values against ψ-values. From the plot it is evident that certain combinations of ϕ and ψ values are frequently encountered while others are never or very rarely encountered.

Many of the amino acids in a protein adopt dihedral angles from seemingly random areas of the Ramachandran plot. These amino acids are commonly referred to as coil residues. Sometimes, however, a number of successive amino acids all adopt dihedral angles within the same, narrow area of the Ramachandran plot, causing them to form special kinds of local structures, referred to as secondary structures.

A number of secondary structures are repeatedly observed in proteins. The most common ones are helices and sheets. There are several different helices, but the α-helix (depicted in Figure 2.4) is by far the most common one. All helices are formed by a number of amino acids that are placed sequentially in the primary sequence. The structure is held firmly in place by hydrogen bonds which form between the backbone C=O-group of amino acid i and backbone N–H-group of amino acid $i + 4$.

β-sheets are formed by two or more β-strands (see Figure 2.5). While a β-strand, like helices, are formed by a number of sequential amino acids, β-sheets are formed by strands that may be very distant in the primary sequence. The β-strand is in itself not a stable structure, but in a β-sheet structure the strands form hydrogen bonds between them which does make them very stable. The strands may pair up in a parallel or anti-parallel fashion as shown in Figure 2.5. Approximately 80% of all β-strands pair up in an anti-parallel manner, making anti-parallel parring significantly more frequent than parallel parring.

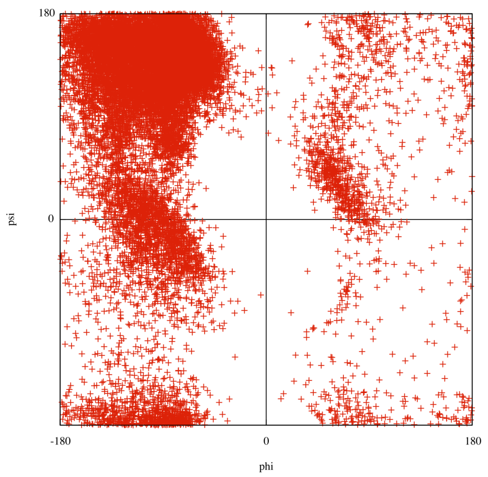


Figure 2.6: A Ramachandran plot of proteins in the PDBSelect25 data set.

2.3 The PSP problem

Essentially the protein structure prediction problem can be stated quite simply as a minimization problem:

> *Given a sequence of amino acids, find the three-dimensional structure with the lowest energy as defined by some energy function.*

We know that a protein will always fold into the same three-dimensional structure under normal conditions, and it will do so very fast – usually within milliseconds. Obviously, proteins do not adopt every possible conformation in order to find the structure with the lowest free energy. This would take far too long. In fact, Levinthal's paradox [10] states that even if a protein only consisted of 100 amino acids that each could take on only two different ϕ and ψ angles, there would be in the order of 10^{30} possible conformations. This means that even if a protein could try 100 billion different conformations per second, it would require 100 billion years – longer than the age of the universe – to try on all possibilities.

Levinthal's paradox has led to the hypothesis of a folding pathway that proteins follow when they fold, but if such a pathway exists it has yet to be discovered. The only pathway-like observations that seem to be clearly observed during protein folding are the formation of secondary structures which usually happens early on in the folding process. A research field within PSP is exclusively concerned with prediction of these secondary structures, and with success rates around 80% secondary structure prediction is generally considered to be quite accurate, and most PSP algorithms include secondary structure predictions [6].

However, aside from inclusion of secondary structure predictions, algorithms can be and are in fact constructed in many different ways. The following section contains a survey of the many recently published PSP *ab initio* algorithms along with a comparison of their reported performance. Only papers that have calculated RMSD (Root Mean Square Deviation) between their test proteins and the native proteins are included. There exists other ways of measuring how close a given protein is to the native protein, but RMSD is the most common one.

2.4 Paper: A Comparative Study of the Reported Performance of *Ab Initio* Protein Structure Prediction Algorithms

This paper was published online in December 2007 (in print April 2008) in the Journal of the Royal Society Interface. It is followed by a supplement that adds a brief discussion of the literature that has appeared since the publication of this paper.

A Comparative Study of the Reported Performance of *Ab Initio* Protein Structure Prediction Algorithms

Glennie Helles

Abstract

Protein structure prediction is one of the major challenges in bioinformatics today. Throughout the past five decades many different algorithmic approaches have been attempted, and while progress has been made the problem remains unsolvable even for many small proteins. While the general objective is to predict the three-dimensional structure from primary sequence, our current knowledge and computational power is simply insufficient to solve a problem of such high complexity.

Some prediction algorithms do, however, appear to perform better than others, although it is not always obvious which ones they are and it is perhaps even less obvious why that is. In this review the reported performance results from 18 different recently published prediction algorithms are compared. Furthermore, the general algorithmic settings most likely responsible for the difference in the reported performance are identified, and the specific settings of each of the 18 prediction algorithms are also compared.

The average normalized RMSD score reported range from 11.17 to 3.48. With a performance measure including both RMSD scores and CPU time, the currently best performing prediction algorithm is identified to be the I-TASSER algorithm. Two of the algorithmic settings – protein representation and fragment assembly – were found to have definite positive influence on the running time and the predicted structures respectively. There thus appear to be a clear benefit from incorporating this knowledge in the design of new prediction algorithms.

1 Introduction

Ram Samudrala once wrote "Proteins don't have a folding problem. It's we humans that do"[37] and indeed that seems to be the case. For five decades researchers all over the world have tried to break the code and predict the

3-dimensional structure of proteins from their primary sequence. There are two very different approaches to protein structure prediction: comparative modeling and *ab initio* prediction.

In comparative modeling predictions are based on knowledge of structures of already known proteins, such that the sequence of an unknown protein is aligned to known proteins and if a homology of more than 35% exists, then the 3-dimensional fold is assumed to be the same [9]. Significant progress has been made in comparative (also called homology) modeling, as the method has proven to be quite efficient and applicable for a majority of proteins [44].

There are, however, three reasons why *ab initio* folding remains interesting. First of all, there still exists a large number of proteins which do not show any homology with proteins of known structure. Secondly, comparative modeling does not offer any insight as to why a protein adopts a certain structure, and thirdly, although some proteins show high resemblance to other proteins they still adopt different structures, which in principle means that predictions made by comparative modeling are never fully reliable.

Many different definitions of *ab initio* algorithms exists. The same definition as in [14] is adopted here, such that the term is taken to mean to start without knowledge of globally similar folds, which allows for algorithms to use statistical information, secondary structure prediction and fragment assembly (refered to by some as *de novo* rather than *ab initio* prediction).

A vast number of *ab initio* algorithms have been proposed throughout the years, with two prominent focus areas, rapidity and quality. Some model only very general principles of protein folding [13, 15, 28], which is fast but typically not very accurate. On the other hand, some create an actual simulation of the folding process [42], which yields excellent results but an unacceptable running time. Most structure prediction algorithms try to balance the two and lie somewhere in between.

This review compares a wide range of *ab initio* protein structure prediction algorithms in order to both identify current state-of-the-art algorithms and to make the effect of algorithm choice and algorithmic configuration stand out. In order to design new and better prediction algorithms it is important to know the paths already traveled, and this review is also meant to facilitate this.

Some of the algorithms proposed have competed in the bi-annual CASP competition (Critical Assessment of Techniques for Protein Structure Prediction)[1], which provides an ultimate way for benchmarking prediction systems. However, the CASP competition is only concerned with quality of the predicted structures. Neither the algorithmic details nor the running time is considered. This review includes all elements that can effect performance. Both algorithms proposed in connection with the CASP free modeling category (for a review of the lastest CASP competition the reader is referred

[1]http://www.predictioncenter.org/

to [17]), and algorithms proposed elsewhere are included if they have been published along with their results within the past five years. Several systems entered in the CASP competition are not published and although the results are available on the Internet, the algorithmic details are unknown, and such systems are therefore excluded. Incidently, the top performing algorithms in the CASP competitions tend to be published, although the SBC system entered by Elofsson and Wallner along with the MQAP-Consensus system from Gattie and the Luethy system are all examples of systems that have appeared to performed very well in the free modeling category of the latest CASP VII competition [2], but to the best of our knowledge are unpublished.

In the next section, key parameters relevant for comparing prediction algorithms are identified, and the following section concerns performance comparison of the reported results from 18 prediction systems. The results of the comparison are discussed in section 4 and our conclusions are summarized in section 5.

2 Algorithmic performance factors

Many parameters influence the running time of a structure prediction algorithm and the quality of the result. In order to determine why some prediction systems are more successful than others, it is important to identify all of the elements that can influence performance. Depending on the problem, some search algorithms (e.g. genetic algorithm or simulated annealing) do for instance tend to perform better than others, and the choice of algorithm may thus influence performance. Some prediction systems utilize the same underlying optimization algorithm but differ in the configuration of the chosen algorithm. In other words, they differ in the setting of algorithmic parameters like protein representation, acceptable angle space and energy function. Furthermore, they composition of the test set may also be responsible for the differences in reported performance.

The effect of one particular setting compared to another is difficult to document based on the results from the literature, because the configuration of individual prediction algorithms typically differ on several settings. Also, some configurations work well for one type of algorithm, but not for other types of algorithms. This is perhaps particularly pronounced between algorithms that employ a multiple solution search strategy – i.e., algorithms that search the entire solution space at once (like genetic algorithms) – and algorithms that employ a single solution search strategy – i.e., algorithms that search neighborhoods (like Monte Carlo). When one thus looks at only a single well-performing algorithm that utilizes a simplified protein representation, it is difficult to know if the good performance has anything to

[2]When looking at RMSD values published on http://www.predictioncenter.org/casp7/Casp7.html

do with the protein representation or if it is really due to for instance the chosen restrictions on angle space or perhaps the chosen set of test proteins. However, if many systems achieve good performance using a specific protein representation, then that protein representation is most likely a good idea. Regardless of the algorithm used, the trick is to introduce restrictions to the search space in a way that yields the proper trade-off where the relative gain in speed does not exceed the relative loss in quality.

Protein structure prediction is highly complex, and restrictions that can decrease the solution space in order to make the problem more tractable are very attractive. The specific configuration of an algorithm reflects the restrictions that are introduced and as mentioned it is usually distinct for every prediction system. The purpose of this review is thus not simply to list the results obtained with the different algorithms, but also to compare the algorithms with respect to the settings in order to better understand why some prediction systems appear to perform better than others.

The following subsections deal with different factors that may affect the performance of the search algorithms used in prediction systems.

2.1 Representation

A protein can be represented in a number of ways ranging from an all-atom to a simple C_α-trace representation. The all-atom model is naturally the most accurate representation, but unfortunately it typically has a very direct negative effect on running time, as more atoms require more time per iteration of the algorithm. Excluding the small hydrogen atoms from the representation and compensating by making the binding atoms larger is a restriction that intuitively seem rather harmless, and it greatly reduces the number of atoms that need to be considered. Further reduction can be made by substituting explicit side chain representation with a single point representing just the center of mass. The CABS (CAlpha,CBeta,Side chain) [22] and the UNRES (UNified RESidue) [27] models are popular examples of this type of reduction. Excluding side chains altogether and thereby including only backbone atoms is yet another simplification that can be made, and at the far end of the scale we have the C_α-trace representation which is no doubt the most optimal representation with respect to running time. Of course it is also a rather crude approximation of the protein.

It should be noted that although these are the types of representations typically encountered in protein structure prediction systems, even cruder simplifications can be made. Experiments with designs of simplified residue alphabets have been made where the amino acids are no longer viewed as distinct, but grouped in categories according to their physical propensities. The best known property-based sequence representation is probably the hydrophobic-hydrophilic alphabet, but many others exist (see for instance [4]). However, such representations are primarily used in model systems rather than real structure prediction systems.

2.2 Dihedral angle space

In principle, an infinite number of angles, dihedral angles and bond lengths between atoms can be adopted, but due to the physical propensities of atoms, certain bond lengths and angles are strongly favored. By analyzing known proteins it is also clear that amino acids have a definite preferences for specific dihedral angles [34]. A very common way to reduce the solution space is thus to fix bond lengths and angles and put restriction on the dihedral angle space by for example incorporating rotamer libraries [8] or operating on lattices.

Many restrictions on the dihedral angle space means that the algorithm will typically converge more quickly than if there are none or only very few restrictions. However, many restrictions on dihedral angle space also means that a significant part of the solution space cannot be sampled, and that the native structure may be unattainable. The dihedral angle space sampled by prediction systems is nearly always restricted in one way or another.

2.3 Energy function

The energy function is probably the parameter that has received the most attention throughout the years, and for good reason, as the energy function has an unquestioned influence on the accuracy of the structures predicted. A rather diverse set of functions that range from very simple to highly complex exists, but a perfect energy function that will consistently identify the native structure among decoys independently of the protein has yet to be found. Simple energy functions are typically based on some very general principles of protein folding such as hydrophobic packing and hydrogen bonding whereas the more complex functions incorporate many other kinds of physical, chemical and statistical information, such as electrostatic potentials, secondary structure tendencies and its like.

Much research has been done in the field of energy functions [39] and a number of major energy functions (also known as force fields) exist, such as CHARMM and AMBER (see for instance [30] for an overview), but most prediction algorithms define and utilize their own versions. A thorough description of the different energy functions is beyond the scope of this study, and the reader is referred to the individual papers for a detailed description of the particular energy function used.

Generally, the energy functions can, however, be divided into two groups: physics-based energy functions and statistics-based energy functions. Physics-based energy functions rely on calculation of energy in the protein, whereas statistics-based energy functions derive their potential from statistical observations. It is important to note that both types of energy functions are approximations, although statistics-based energy functions are perhaps generally considered the cruder approximation of the two.

2.4 Folding strategy

Predicting the structure of small proteins is naturally easier (although still hard) than predicting the structure of large proteins, since the solution space grows exponentially with the number of amino acids. This fact has motivated many to divide the proteins into a number of fragments whose structure is predicted separately and subsequently assembled - a strategy known as fragment assembly. The method is currently very popular and also very successful when either secondary structure prediction algorithms (like PSIPRED [18]) or fragment libraries are used to predict fragment structures.

However, two things should be noted in this respect. First of all, by using fragments one assumes that a fragment always folds into a number of predefined ways. Once a fragment is selected it is considered rigid and it can only be replaced by another fragment. Long range interactions in the protein under investigation are therefore not directly involved in shaping the fragments, which may not be prudent. Secondly, although algorithms that are based on secondary structure prediction and/or fragment libraries are currently better at generating native-like structures, they are, of course, forever bound to the limitation of systems relying on known structures, as secondary structure prediction algorithms are trained on known structures and fragment libraries built from known structures.

2.5 Test set

Unfortunately, a standard protein test set does not exist, and so yet another parameter that must be considered when evaluating the performance of a prediction algorithm, is the set of test proteins. The number of proteins in the test set is of interest for statistical reasons. For a large test set it is less likely that good results are obtained by mere “luck” and the algorithm is more likely to be generally applicable than if the test set contained only a few test proteins. Since most prediction algorithms are quite time consuming, test sets are most often of limited size (≤ 15), although the largest test set seen in this study includes 125 test proteins [43].

The lengths of the individual test proteins (e.g., number of amino acids) are of also interest, as the structures of small proteins are both easier and faster to predict than the structures of larger proteins. Skolnick et al. [35] have previously investigated the possibility of obtaining a native-like structure by mere chance, and concluded that generating a structure by random (although compact) that has a RMSD below 6Å is highly unlikely for a chain greater than 60 amino acids, but naturally that chance increases for smaller proteins. An algorithm that is able to predict the structure of a protein of, say, 20 amino acids to an RMSD of 6Å is much less impressive than an algorithm that can predict the structure of a protein of, say, 100 amino acids to an RMSD of 6Å. Most test proteins contain less than 100 amino acids.

Finally, the structural classes of the test proteins are of interest. All proteins can generally be classified as either α, β, α/β or coil [31], where the first three categories are by far the most populated. A good prediction algorithm must be able to make equally good predictions regardless of structural class. Hence confidence in an algorithm relies also on the structural classes of the test proteins. It is much harder to conclude anything about the general applicability of an algorithm that has only been tested on a few proteins belonging to the same structural class than if the algorithm has been tested on a larger number of proteins from all structural classes. Most – but not all – prediction systems are tested on proteins from all structural classes (except coil-structures which none of the systems included in this study are tested on).

3 Performance comparison

When configuring an algorithm for structure prediction, focus can be put on any or all of the parameters identified in the previous section, which is reflected in the numerous prediction algorithms proposed. The aim of this performance comparison is primarily to contrast different algorithmic approaches, but also to deduce any trends in the settings of the algorithms. One must, of course, be cautious when drawing conclusions about a given setting, as it is often tied to the algorithm and the test proteins, but when comparing a relatively large number of algorithms, the results may nevertheless indicate some general trends that would be of interest in the design of new algorithms.

Performance is here compared between the reported results of 18 recently proposed prediction algorithms (published in the last 5 years). Several algorithms have been excluded (even relatively known *ab initio* systems such as [19] and [45]) because RMSD values between the native and predicted structures have not been published in their papers. There exist a number of alternative ways to compare structures (such as dRMSD, GDT_TS, TM score, etc.), and while they may be better at expressing how well the algorithm performs in terms of for example substructure formation or core packing, RMSD is the most commonly used descriptor. Furthermore, an overall low RMSD is the ultimate goal for a structure prediction algorithm if it is to be used in practice. Algorithms designed to predict only specific types of proteins (like for example membrane proteins) are excluded. Algorithms based on exact knowledge of the native structure are naturally also excluded, although the contribution from [7] is very interesting from an algorithmic point of view. Finally, newer versions of the algorithms are assumed to be equally good or better than older versions, and therefore only the latest versions have been included.

Although two of the algorithms participated in the CASP VII competition [3, 40], their results from the competition are not included. As mentioned earlier, RMSD values are available on the internet, but have not yet been explicitly documented in the literature.

3.1 Results

The result of the performance comparison is presented in Table 1. The three columns 'Avg. RMSD', 'Res. set size' and 'Running time' constitutes the collected results for each algorithm.

The 'Avg. RMSD' column specifies the average RMSD values for the best selected structures of all the test proteins. The 'Res. set size' refers to the size of the result set, i.e. the number of predicted structures selected by the systems. For those systems that use clustering or refinement it refers to the number of results selected after the initial results have been clustered or refined. Many are reluctant to pick one cluster over another and thus return a representative structure from each cluster. There may be significant differences between the representative structures chosen (see for instance [40]), but RMSD is usually not reported for all selected structures and thus only the lowest RMSD value amongst the selected structures is included in the RMSD average here.

The 'Running time' column indicates how quickly the algorithm finds a solution for a protein. Running times generally depend on the length of the proteins, but as mentioned earlier – and as evident from Table 1 – most test proteins included in the test sets are of comparable length. The computer power available differs greatly, but the time stated in the column gives a rough estimate of the time required for a single standard PC processor to reach a solution. Hence, if a group has used a cluster of 10 computers and 4 days to predict a structure, it will be marked as 'months' ($\approx$ 40 CPU days). The algorithms (and results) included have all been published in the last 5 years; while that is a fairly short time range, it should be noted that computer power has increased significantly in that period (a standard PC is roughly 3 times faster today), and newer algorithms have thus not only the benefit of previous experience, but also the benefit of significantly faster computers.

	Rep.[a]	Frag.[b]	Dihedral space[c]	Energy func.[d]	Running time[e]	# of proteins[f]	Length range[g]	Class[h]	Res. set size[i]	avg. RMSD[j]
- *MD simulation*										
Pande group [42]	All-atom	No	All	Physics	Years	1	[36]	α	1	1.7
Liwo group [29][k]	UNRES	No	All	Physics	Hours	4	[46-75]	α	10	2.6
						1	[48]	α/β	10	3.9
- *Replica Exchange Monte Carlo*										
Shakhnovich group [41]	All-atom	No	All	Statistics	Weeks	7	[46-77]	α	15	4.3
						3	[24-67]	β	15	5.1
						3	[57-75]	α/β	15	4.4
Touchstone II [43]	CABS	Yes	Lattice	Statistics	Days	43	[36-157]	α	2-22	6.7
						41	[39-153]	β	4-62	8.2
						41	[47-174]	α/β	2-56	8.3
Kolinski group [24]	CABS	Yes	Lattice	Statistics	Days	4	[66-86]	α	1	4.8
						3	[45-137]	β	1	8.5
						2	[56-76]	α/β	1	2.6
ZAM [32]	All-atom	Yes	Restricted	Physics	Months	3	[46-73]	α	1	3.0
						2	[25-26]	β	1	2.1
						4	[35-57]	α/β	1	3.0
- *Metropolis Monte Carlo*										
Rosetta [3, 36]	All-atom	Yes	Restricted	Statistics	Months	3	[49-72]	α	5	1.5
						2	[59-67]	β	5	8.0
						11	[59-88]	α/β	5	3.6
- *Monte Carlo Simulated Annealing*										
SimFold [11]	CABS-like	Yes	Restricted	Mixed	Months	14	[63-109]	α	400	3.9
						10	[28-72]	β	400	7.5
						12	[28-72]	α/β	400	6.6
PROTINFO [16]	All-atom	No	Restricted	Statistics	Months	1	[53]	α	5	5.0
						1	[69]	α/β	5	4.3
- *Hyperbolic Monte Carlo*										
I-TASSER [40]	C_α	Yes	Part lattice	Statistics	Hours	16	[49-118]	α	5	3.5
						14	[47-117]	β	5	3.8
						26	[47-117]	α/β	5	4.3
- *Conformational Space Annealing (CSA)*										
Profesy [25, 26]	Backbone	Yes	Restricted	Physics		2	[28-46]	α	5	4.2
						2	[20-57]	β	5	2.7
- *Multi-objective Evolutionary approach*										
Nicosia [6]	All-atom	No	Restricted	Physics	Hours[l]	3	[34-70]	α	1	3.3
						1	[46]	α/β	1	4.4

	Rep.[a]	Frag.[b]	Dihedral space[c]	Energy func.[d]	Running time[e]	# of proteins[f]	Length range[g]	Class[h]	Res. set size[i]	avg. RMSD[j]
- *Evolutionary algorithms*										
Schug [38]	% H	No	Restricted	Physics	Hours	1	[60]	α	10	3.9
Profet [23]	All-atom	No	All	Physics	Days	1	[34]	α	5-10	1.6
						2	[18 -21]	β	5-10	5.2
- *MOLS-Genetic algorithms*										
Gautham [1]	All-atom	Yes	Restricted	Physics	Hours	4	[26-52]	α	1	4.9
						1	[16]	β	1	1.6
- *Branch and Bound*										
Paluszewski [33]	C_α	Yes	Restricted	Statistics	Days	3	[43-65]	α	10000	4.6
						1	[48]	β	10000	5.0
						2	[68-76]	α/β	10000	6.3
- *αBB/CSA/MD*										
ASTROFOLD [21]	All-atom	Yes	Restricted	Physics	Weeks	1	[69]	α	1	6.2
						3	[75-101]	β	1	5.6
						4	[56-95]	α/β	1	4.9
- *Probabilistic distance geometry optimization*										
PROPAINOR [20]	C_α	Yes	All	Statistics	Hours	21	[70-130]	α	5-15	5.6
						13	[74-130]	β	5-15	6.9
						18	[76-129]	α/β	5-15	7.2

[a]The protein representation applied.
[b]Fragment assembly.
[c]Possible dihedral angles.
[d]Energy function applied. Energy functions are either statistics or physics based.
[e]A rough estimate of how quickly a solution is found using a single processor.
[f]Number of test proteins.
[g]Sequence length range of the test proteins.
[h]Structural class of the test proteins.
[i]The number of solutions returned by the systems (after clustering), from which the lowest RMSD structure is picked.
[j]The average RMSD values of the best selected structures found for all test proteins belonging to this structural class.
[k]The algorithm did not always converge (betasheet and alpha/betasheet).
[l]From personal correspondance.

Table 1: Performance comparison of 18 structure prediction algorithms.

4 Discussion

4.1 Algorithmic configuration

Detailed MD simulation of all-atom protein models such as [42] is typically performed in order to allow researchers to observe the folding pathways – not merely to predict protein structure. MD simulation has, however, proved to be very accurate (for at least small proteins), and it has been included here because it is technically possible to use MD simulation for structure prediction, although it is computationally extremely heavy and renders the problem intractable for all but the smallest of proteins. In a sense one might say that the goal of a prediction algorithm is to combine the accuracy of MD simulation with the speed of (most) search algorithms.

Different flavors of the Monte Carlo (MC) search strategy are by far the most common types of algorithms used, but results for all algorithms are for the most part comparable with respect to accuracy. Studies that compare different MC search strategies are performed regularly [12]. They typically show that one algorithm performs slightly better than the ones it is compared to, but this may be related to the test proteins rather than the algorithm as such. From Table 1 it would certainly seem like all types of MC searches show roughly the same performance. The I-TASSER algorithm [40] based on a Hyperbolic Monte Carlo scheme stands out with results being superior to the others, particularly when it comes to running time.

Interestingly, excellent results are obtained quickly by the Liwo group [29] who also use MD simulation but with a simplified residue representation. The force field used is designed to compensate for the missing atoms, and while some simulations – particularly of proteins containing β-sheets – do not converge to a final structure, the fact that the algorithm reached good results extremely fast indicates an enthralling potential. In fact, simplified residue representation appears to generally have a positive effect on the running time of the algorithms but a more or less undetectable effect on accuracy irrespectively of the type of algorithm used. As mentioned previously, most prediction algorithms differ on multiple settings, and it is thus usually difficult to make any general conclusions about a particular setting. Nevertheless, in this case where many algorithms are aligned, it seems clear that the effect of a simplified protein representation on overall accuracy is minimal.

Algorithms based on fragment assembly are generally considered more successful than others (many of the best algorithms in the CASP competition use this folding strategy), and it certainly seems intuitively right that fragment assembly would be much faster. This is not evident from the results reviewed here where the strategy does not appear to have any major influence on running time. However, fragment assembly is most likely an important factor in the high accuracy achieved by algorithms such as I-TASSER and

Rosetta, but one should also bear in mind that the use of fragments makes the system a borderline "comparative modeling"-system, which relies heavily on existing structures.

Aside from MD simulation, the dihedral angle space is restricted in nearly every algorithm. PROPAINOR stands out as it takes a completely different approach to the problem by sampling residue distance space rather than angle space – thereby making all dihedral angles possible. Of course, the restrictions they put on residue distances for a given protein may in fact translate into restrictions on dihedral angle space for that particular protein, but a common set of restrictions are not build into the algorithm. Most algorithms are off-lattice, but some make use of a lattice (at least for parts of the protein) and with an expected positive influence on at least the running time [24, 40, 43]. It should also be noted that the restrictions on dihedral angle space used by most do not appear to have a detectable influence on the quality of the predicted structures.

With regards to the two types of energy functions it would appear from this comparison that algorithms that utilize a physics-based energy function find solutions that are marginally better than the solutions found by algorithms with a statistics-based energy function. No influence on the running time of the algorithms can be observed. It should be noted that statistics-based energy functions vary greatly in the number of parameters they include and thus a general trend should not be extracted from this study.

4.2 Test set specific parameters

From Table 1 it can be seen that only 11 of the 18 algorithms have been tested on proteins from all structural classes (except coil). The Liwo group [29] did in fact test on proteins from the three main structural classes, but as the algorithm did not converge for β-sheet structures, they have not reported any results for these proteins. The smallest test sets used included only one structure [38, 42] while the largest set included 125 structures.

Surprisingly, there appears to be no correlation between running time and quality – in fact the fastest algorithms obtain some of the best results even when the lengths of the proteins are taken into consideration. Algorithms that have been tested on test sets, which include many and/or significantly larger proteins, would be expected to obtain a higher average RMSD, but the results would also be more reliable as it is very difficult to tune parameters – intensionally or not – to produce good results on large test sets (as discussed in section 2). TOUCHSTONE II (large test set) and to a certain extent PROPAINOR (large proteins) support this assumption, but the I-TASSER algorithm actually maintains excellent performance despite being tested on the second largest test set with proteins that are both structurally diverse and have an average length of 81 amino acids (Table 2).

From the results presented, it is evident that predicting β-sheets is much

	# proteins [a]	Avg. length[b]	Avg. RMSD[c]	Avg. RMSD_{100}[d]	Run time[e]
Pande group [42]	1	36	1.7	3.48	Years
Liwo group [29]	6	57	4.0	5.56	Hours
Shakhnovich group [41]	13	58	4.5	6.18	Weeks
Touchstone II [43]	125	87	7.7	8.28	Days
Kolinski group [24]	9	78	6.2	7.08	Days
ZAM [32]	9	46	2.8	4.57	Months
Rosetta [2, 36]	16	69	3.8	4.66	Months
SimFold [11]	38	76	5.9	6.84	Months
PROTINFO [16]	2	61	4.7	6.24	Months
I-TASSER [40]	**56**	**81**	**4.2**	**4.69**	**Hours**
Profesy [25, 26]	4	38	3.5	6.78	
Nicosia [6]	4	52	3.5	5.20	Hours
Schug [38]	1	60	3.9	5.24	Hours
Profet [23]	3	28	4.0	11.17	Days
Gautham [1]	5	33	4.2	9.42	Hours
Paluszewski [33]	6	60	5.2	6.96	Days
ASTROFOLD [21]	8	76	5.3	6.15	Weeks
PROPAINOR [20]	52	98	6.4	6.47	Hours

[a]Number of test proteins.
[b]Average length of the test proteins.
[c]Average RMSD values of the best selected structures found for all test proteins.
[d]Average RMSD values are normalized with respect to protein lengths.
[e]A rough estimate of how quickly a solution is found using a single processor.

Table 2: Summarized results

more difficult, and most algorithms that are tested on proteins belonging to different structural classes perform worse on proteins that contain β-sheets, indicating that the energy function used is biased toward one kind of secondary structure (usually α-helices). The results reported for the ZAM [32] and Profesy [25, 26] algorithms along with the results reported by Gautham [1] actually showed better results for β-class structures, but that is most likely due to the short length of the selected β-class proteins. A few of the groups [20, 21, 40, 41] stand out as they seem to obtain equally good results for all their proteins regardless of structural class.

4.3 Performance results

Generally, most results reported look impressive. It should of course be emphasized that the RMSD values reported here are for the selected structures with the lowest RMSD values found by the prediction systems. Note that most algorithms return numerous structures, some with high RMSD values and some with low RMSD values. A large result set size is generally less attractive – even if it includes a near native structure – because the algorithm as such is unable to separate that structure from the decoys. Returning many solutions does, however, not necessarily pose a problem, if there is some way to separate the 'good' structures from the 'bad' by for example using clustering [41] or other filtering techniques [10]. As shown by the 'Res. set size' column in Table 1 many systems return several solutions even after the results have been clustered and the best solution is then picked based on its RMSD score to the native protein. Of course, in order to function as a reliable prediction system, one must be able to pick the good structure without knowledge of the native structure.

Table 2 summarizes the performance result of the predictions. In the 'Avg. RMSD_{100}' column, the average RMSD values have been normalized with respect to the length of the test proteins [5], which makes it easier to compare RMSD values for proteins of different lengths. Most of the included published results have an average RMSD_{100} value around 6Å. Although the result reported by the Pande group using the MD simulation is the lowest, it has two major draw backs: there are too few test proteins and the running time is very poor. Furthermore, the protein folded is very small (only 36 amino acids). From the RMSD_{100} values it is clear that the Rosetta algorithm [3] and the I-TASSER algorithm [40] are at a near tie, which was also seen in the latest CASP VII competition [17]. The Rosetta algorithm [3] holds a long standing record for achieving good results at the CASP competitions, and so the results of the 16 test proteins is considered quite reliable. The I-TASSER algorithm is, however, tested on a much larger test set of 56 proteins (including the same proteins as Rosetta was tested on), and with an excellent running time that clearly outperforms Rosetta, it is here concluded to be the overall best performing algorithm. As mentioned previously, the CASP VII results from I-TASSER and Rosetta are available on the internet, but not included here. However, analysis of the RMSD values from CASP VII reveals a picture similar to what is seen here. Both I-TASSER and Rosetta use a MC sampling scheme (although different variants), fragment assembly and a statistics-based energy function, but they differ in protein representation and acceptable dihedral angle space.

Finally, the need for a standard protein test set of appropriate size must be emphasized. The trends observed concerning parameter settings in this study are based on (sparse) statistics, but could perhaps be made into actual conclusions if all research groups used the same test set (as it is seen in other research areas). Furthermore, a standard test set would make it difficult to cheat and would allow for a more systematic and reliable evaluation of algorithms.

5 Conclusion

The parameters for proper comparison of protein structure prediction algorithms have been identified and the performance of 18 different *ab initio* prediction algorithms have been compared with respect to these parameters.

In lack of a standard protein test set, it is usually difficult to evaluate the importance of one particular parameter setting over another, but because of the relatively large number of algorithms compared here, certain trends in parameters settings could be identified. Simplified protein representation was found to have seemingly undetectable influence on accuracy, but a definite positive influence on running time. The (very popular) fragment assembly folding strategy is most likely responsible for the high accuracy achieved by some groups [3, 40], but it does not appear to have any general influence on running time. Half of the algorithms use a physics-based energy function and although they appear to slightly outperform those utilizing a statistics-based energy function, the complexity of energy functions makes it impossible to draw any reliable conclusions about the effect of physics-based versus statistics-based energy functions. Surprisingly, the overall best performing algorithm – the I-TASSER algorithm [40] – is also one of the fastest algorithm included in this study.

As a final note, it should be mentioned that this type of performance comparison is made particularly difficult because research groups test their algorithms on their own selected proteins. A standard protein test set would greatly enhance any possible trends in parameter settings and could facilitate designs of new algorithms.

References

[1] J. Arunachalam, V. Kanagasabai, and N. Gautham. Protein Structure Prediction using mutually orthogonal Latin squares and a genetic algorithm. *Biochem. Biophys. Res. Com.*, 342:424–433, 2006.

[2] P. Bradley, L. Malmstrœm, B. Qian, J. Schonbrun, D. Chivian, D. E. Kim, J. Meiler, K. M. S. Misura, and D. Baker. Free modeling with Rosetta in CASP6. *Proteins*, 7:128–134, 2005.

[3] P. Bradley, K. M. S. Misura, and D. Baker. Towards High-Resolution de Novo Structure Prediction for Small Proteins. *Science*, 309:1868–1871, 2005.

[4] A. C. Camproux and P. Tuffery. Hidden Markov model-derived structural alphabet for proteins: the learning of protein local shapes captures sequence specificity. *Biochem. Biophys. Acta*, 1724:394–403, 2005.

[5] O. Carugo and S. Pongor. A normalized root-mean-spuare distance for comparing protein three-dimensional structures. *Protein Sci.*, 10:1470–1473, 2001.

[6] V. Cutello, G. Narzisi, and G. Nicosia. A multi-objective evolutionary approach to the protein structure prediction problem. *J. R. Soc. Interface*, 3:139–151, 2006.

[7] K. W. DeRonne and G. Karypis. Effective optimization algorithms for fragment-assembly based protein structure prediction. *Comp. Sys. Bioinf. Conf.*, pages 19–29, 2006.

[8] R. L. Dunbrack. Rotamer libraries in the 21st century. *Curr. Op. Struct. Biol.*, 12:431–440, 2002.

[9] Y. J. K. Edwards and A. Cottage. Bioinformatics Methods to Predict Protein Structure and Function. *Mol. Biotech.*, 23:139–166, 2003.

[10] E. Eyal, M. Frenkel-Morgenstern, V. Sobolev, and S. Pietrokovski. A pair-to-pair amino acids substitution matrix and its applications for protein structure prediction. *Proteins*, 67:142–153, 2007.

[11] Y. Fujitsuka, G. Chikenji, and S. Takada. SimFold Energy Function for De Novo Protein Structure Prediction: Consensus with Rosetta. *Proteins*, 62:381–398, 2006.

[12] D. Gront, A. Kolinski, and J. Skolnick. Comparison of three Monte Carlo conformational search strategies for a proteinlike homopolymer model: Folding thermodynamics ad identification of low-energy structures. *J. Chem. Phy.*, 113:5065–5071, 2000.

[13] Y. Z. Guo, E. M. Feng, and Y. Wang. Optimal HP configurations of proteins by combining local search with elastic net algorithm. *J. Biochem. Biophys. Met.*, 70:335–40, 2007.

[14] C. Hardin, T. V. Pogorelov, and Z. Luthey-Schulten. *Ab initio* protein structure prediction. *Cur. Op. Struct. Biol.*, 12:176–181, 2002.

[15] J. Hockenmaier, A. K. Joshi, and K. A. Dill. Routes are trees: the parsing perspective on protein folding. *Proteins*, 66:1–15, 2007.

[16] L. Hung, S. Ngan, T. Liu, and R. Samudrala. PROTINFO: new algorithms for enhanced protein structure predictions. *Nuc. Acids Res.*, 33:Online, 2005.

[17] R. Jauch, H. C. C. Yeo, P. R. Kolatkar, and N. D. Clarke. Assessment of CASP7 structure predictions for template free targets. *Proteins*, online, 2007.

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

[19] D. T. Jones, K. Bryson, A. Coleman, L. J. McGuffin, M. I. Sadowski, J. S. Sodhi, and J. J. Ward. Prediction of Novel and Analogous Folds Using Fragment Assembly and Fold Recognition. *Proteins*, 7:143–151, 2005.

[20] R. R. Joshi and S. Jyothi. Ab-initio prediction and reliability of protein structural genomics by PROPAINOR algorithm. *Comp. Bio. Chem.*, 27:241–252, 2003.

[21] 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. Jour.*, 85:2119–2146, 2003.

[22] A. Kolinski. Protein modeling and structure prediction with a reduced representation. *Acta Biochimica Polonica*, 51:349–371, 2004.

[23] F. Koskowski and B. Hartke. Towards Protein Folding with Evolutionary Techniques. *J. Comp. Chem*, 26:1169–1179, 2004.

[24] D. Latek, D. Ekonomiuk, and A. Kolinski. Protein Structure Prediction: Combining De Novo Modeling with Sparse Experimental Data. *Wiley InterScience*, online, 2007.

[25] J. Lee, S. Kim, K. Joo, I. Kim, and J. Lee. Prediction of Protein Tertiaru Structure Using PROFESY, a Novel Method Based on Fragment Assembly and Conformational Space Annealing. *Proteins*, 56:704–714, 2004.

[26] J. Lee, S. Kim, and J. Lee. Protein structure prediction based on fragment assembly and parameter optimization. *Biophys. chem.*, 115:209–214, 2005.

[27] J. Lee, A. Liwo, and H. A. Scheraga. Energy-based de novo protein folding by conformational space annealing and an off-lattice united-residue force field: Application to the 10-55 fragment of staphylococcal protein A and to apo calbindin D9K. *PNAS*, 96:2025–2030, 1999.

[28] Z. Li, X. Zhang, and L. Chen. Unique optimal foldings of proteins on a triangular lattice. *Appl. Bioinf.*, 4:105–16, 2005.

[29] A. Liwo, M. Khalili, and H. A. Scheraga. *Ab initio* simulations of protein-folding pathways by molecular dynamics with the united-residue model of polypeptide chains. *PNAS*, 102:2362–2367, 2005.

[30] A. D. MacKerell. Empirical Force Fields for Biological Macromolecules: Overview and Issues. *J. Comp. Chem.*, 25:1584–1604, 2004.

[31] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton. CATH–a hierarchic classification of protein domain structures. *Structure*, 5:1093–1108, 1997.

[32] S. B. Ozkan, G. A. Wu, J. D. Chodera, and K. A. Dill. Protein folding by zipping and assembly. *PNAS*, 104:11987–11992, 2007.

[33] M. Paluszewski and P. Winter. Branch and Bound Algorithm for Protein Structure Prediction Uning Efficient Bounding. *Unpublised*, 00:00, 2007.

[34] G. N. Ramachandran and V. Sasisekharan. Conformations of polypeptides and proteins. *Adv. Prot. Chem.*, 23:283–437, 1968.

[35] B. A. Reva, A. V. Finkelstein, and J. Skolnick. What is the probability of a chance prediction of a protein structure with an rmsd of 6 A. *Fold. Des.*, 3:141–147, 1998.

[36] C. Rohl, C. E. M. Strauss, K. M. S. Misura, and D. Baker. Protein Structure Prediction Using Rosetta. *Met. Enzym.*, 383:66–93, 2004.

[37] R. Samudrala. *Genes, Macromolecules & Computers* . http://www.ram.org/ramblings/dream/, 1990.

[38] A. Schug and W. Wenzel. An Evolutionary Strategy for All-Atom Folding of the 60-Amino-Acid Bacterial Ribosomal Protein L20. *Biophys. Jour.*, 90:4273–4280, 2006.

[39] J. Skolnick. In quest of an empirical potential for protein structure prediction . *Curr. Op. Struct. Biol.*, 16:166–71, 2006.

[40] S. Wu, J. Skolnick, and Y. Zhang. Ab initio modeling of small proteins by iterative TASSER simulations. *BMC Biol.*, 5:Online, 2007.

[41] J. S. Yang, W. W. Chen, J. Skolnick, and E. I. Shakhnovich. All-Atom Ab Initio Folding of a Diverse Set of Proteins. *Structure*, 15:53–63, 2006.

[42] B. Zagrovic, C. D. Snow, M. R. Shirts, and V.S.Pande. Simulation of Folding of a Small Alpha-helical Protein in Atomistic Detail using Worldwide-distributed Computing. *J. Mol. Biol.*, 323:927–937, 2002.

[43] Y. Zhang, A. Kolinski, and J. Skolnick. TOUCHSTONE II: A New Approach to Ab Initio Protein Structure Prediction. *Biophys. Jour*, 85:1145–1164, 2003.

[44] Y. Zhang and J. Skolnick. The protein structure prediction problem could be solved using the current PDB library. *PNAS*, 102:1029–1034, 2004.

[45] H. Zhou and J. Skolnick. Ab initio protein structure prediction using chunk-TASSER. *Biophys J.*, Epub, 2007.

2.5 Postscript

Since the publication of the paper in 2008, there seems to have been a shift of focus in the PSP society. Devising new *ab initio* prediction algorithms for large scale structure prediction appears to have given way to devising algorithms specialized for structure prediction of fragments, short loops, peptides or mini-proteins [3, 8, 9, 11, 12]. The reported performance of these algorithms cannot be fairly compared with the performance of the algorithms presented in [6], as both RMSD and runtime are expectedly better for short fragments. To this date, the i-Tasser and Rosetta algorithm thus remain the best performing algorithms.

Chapter 3

Meta-heuristic Search in PSP

3.1 Background

The PSP problem has been shown to be NP-hard [2], and we thus cannot expect to find a polynomial time algorithm for this problem unless $P = NP$. Like for many other NP-hard problems, most attempts to find the native structure involve the use of some meta-heuristic. Meta-heuristics are typically inspired by nature. Whether it be evolution, cooling of molecules, dancing bees, food-gathering ants or the flocking behavior of some animals we seem to be able to extract some governing principles that can assist us in various optimization problems. The general idea behind these nature-inspired algorithms is to utilize the fact that nature always seems to find the best and most appropriate or efficient way of doing things. As Fred Gratzon, author of the book "The Lazy Way to Success" [5] said:

> *Hard Work is Passè. The paradigm-shifting concept is 'Smart Laziness' - where success comes through cleverly avoiding work but still getting the job done. In this oasis, we celebrate those magical ways where doing less accomplishes more.*

A number of meta-heuristics for general purpose applications have been proposed over the years. The most well-known and frequently used meta-heuristics are probably genetic algorithms and simulated annealing, but new variants surface on a regular basis. Common for all meta-heuristics is their ability to navigate around (infinitely) large solution spaces and find good solutions relatively fast. Unlike exact methods, there is no guarantee that the solution is at a global optimum or even that it is not at a global optimum, but in practice meta-heuristics are able to locate better solutions than exact methods. In fact, exact methods are very rarely encountered at all in this research field.

Creating a good search algorithm is of course of vital importance for a complex problem like PSP, but investigations into performance differences between different meta-heuristics for PSP have been paid surprisingly little attention. For meta-heuristics it is largely the hyper-parameters that are responsible for how efficient the algorithm turns out to be. It could thus be argued that if we use the same hyper-parameters for all meta-heuristics they will do equally well, which is very much in keeping with the “no-free-lunch” theorem that states that all optimization algorithms will on average do equally well over the set of all mathematically possible problems. The problem here is, of course, that there may be significant performance differences within a specific area, and although some hyper-parameter counterparts can easily be identified across different meta-heuristics, others are unique in their nature. A mutation strategy from a genetic algorithm, for instance, can readily be used to control alterations in a simulated annealing procedure, but the cross-over operator, on the other hand, has no counterpart in simulated annealing. One thus cannot assume that the choice of meta-heuristic is irrelevant.

3.2 Parallelism

Parallel meta-heuristics have been known for decades, but they do not seem to have been fully integrated in the PSP field yet. Although attempts have certainly been made, most PSP methods still rely on sequential meta-heuristics.

Essentially, parallel meta-heuristics are a specific class of meta-heuristics that are designed to run in parallel. Many instances or re-runs of the same sequential meta-heuristic can, of course, always be executed simultaneously, but in parallel meta-heuristics parallelism is built into the design. The niche genetic algorithm and the parallel tempering algorithm are two examples of parallel meta-heuristics that are explored in much further detail in this chapter. The interesting thing about parallel meta-heuristics is that they are not only much faster, they also improve solution quality by a factor that vastly exceeds what would be expected from running multiple instances of the sequential algorithm.

3.3 Paper: Exploring Parallel Meta-heuristics for Protein Structure Prediction

This paper is written as a technical report. It presents and compares performance of different meta-heuristics, sequential as well as parallel, for PSP. It focuses specifically on the role of parallelism and demonstrates how parallelism can and should change the way we construct PSP algorithms. A short paper summarizing the main results has been presented at the BIOINFORMATICS 2010 conference and published in the conference proceedings. The short paper is included as Appendix A.

Exploring Parallel Metaheuristics for Protein Structure Prediction

Glennie Helles

Abstract

For many high-complexity problems exhaustive search is infeasible and researchers thus rely on meta-heuristics to solve the problems in a satisfactory way. The problem of predicting the structure of proteins from their primary sequence is NP hard and falls into this category. Looking at the literature, the most frequently applied meta-heuristics for protein structure prediction are without a doubt simulated annealing (or a variant hereof) followed by genetic algorithms. Aside from the terminological advantages of simulated annealing for this particular problem, there are no obvious reasons why simulated annealing would be preferred over genetic algorithms. Although simulated annealing, as the only meta-heuristic, enjoys the luxury of a theoretical guarantee of finding the global optimum, it is well known that this theoretical guarantee bears little importance in practice, since the guarantee only holds given infinite time.

Both simulated annealing and genetic algorithms exist in parallel versions, but neither seems to have gained any real popularity in the field of protein structure prediction. This is despite the fact that the parallel versions have been shown to outperform their sequential counterparts for many other problems. In this report, we describe these two most prominent meta-heuristics and investigate how they compare to each other in the field of protein structure prediction. Focus is given especially to the parallel versions of these algorithms, and we propose a special version of the niche genetic algorithm and demonstrate how low energy structures can be obtained much faster than with any of the other algorithms - both parallel and sequential. The algorithm is highly elitist which would normally set off the exploitation-exploration balance, but in the proposed algorithm, parallelism is specifically used to ensure that the balance is maintained. Using parallelism is a very attractive feature, as multi-core processors and clusters are becoming readily available to most researchers and the "cost" of parallelization is thus close to nothing.

1 Introduction

Meta-heuristics are known to perform well on high-complexity problems where the search space becomes too big for exhaustive search to be feasible. Prediction of the three-dimensional structure of proteins from their primary sequence alone, known as *ab initio* or *de novo* folding [1], is such a problem, and meta-heuristics are nearly always used in attempts to solve this problem [11, 21].

Proteins are made up by amino acids that are strung together like pearls on a string, such that each amino acid is connected to its neighboring amino acid(s) via a peptide bond. The peptide bond is very rigid and the dihedral angle – denoted ω – around this bond is quite rigid. The dihedral angles around the two other backbone bonds, the N–C_α bond and the C_α–C' bond, are denoted ϕ and ψ respectively. Atoms can in theory rotate freely around these two bonds which means that, just like a pearl necklace, a protein can be folded up in infinitely many ways, which is the reason that protein structure prediction poses such a big problem. Fortunately, steric clashes between atoms in neighboring amino acids do impose a considerable restraint on the flexibility of the ϕ and ψ angles actually observed for amino acids [23], but searching exhaustively for the structure with the lowest energy remains intractable.

Judging from the literature, the Monte Carlo variant known as Simulated Annealing appears to be the preferred meta-heuristic for *ab initio* structure prediction with Genetic Algorithms [11] as the second most popular choice. Both meta-heuristics can be parallelized, and the parallel versions are generally believed to perform better in rugged energy landscapes like those associated with protein structure prediction [7]. Oddly enough, the parallel versions are nevertheless used to a much lesser extent than their sequential counterparts in the field of protein structure prediction. We speculate that that is mostly because the effect of the choice of meta-heuristics has been paid little attention in this field that is notoriously haunted by many other fundamental problems. The most significant obstacle is probably to find an appropriate energy function that can be used to score a protein, which has by far received the most attention over the years.

While measuring how close a predicted protein is to the native protein can easily be done by various distance measurements like RMSD (Root Mean Square Deviation) or GDT (Global Distance Test), these measurements naturally cannot be used to score the quality of structures during the search procedure, where

[1]The term *ab initio* traditionally refers to prediction methods that start without any knowledge of any globally similar folds, thereby setting them aside from homology modeling techniques. However, many so called *ab initio* methods do in fact use secondary structure prediction algorithms that are trained from knowledge of already known structures, or they use fragment assembly compiled from known structures. Some choose to refer to this as *de novo* prediction rather than *ab initio* prediction. The term *ab initio* will be used in this publication

the native protein is unknown. Instead, we have to come up with some other way of measuring how close a given structure is to the native state without actually knowing the native state. Myriads of energy functions that attempt to achieve this have been proposed, and most research groups utilize their own version. Some energy functions, the physics-based energy functions, strive to calculate an approximation of the actual free energy of a structure, because the native structure is believed to be have the lowest free energy. Statistics-based energy functions, on the other hand, calculate pseudo-energies based on statistical information about known structures.

In this report, we investigate meta-heuristic performance in protein structure prediction, focusing especially on the parallel versions of simulated annealing and genetic algorithms. In particular, we explore how parallelism can be brought into the design of a parallel genetic algorithm as a hyper-parameter on equal terms as selection strategies, population size and mutation frequencies. The GA variant, which I dub inGA (iterative niche genetic algorithm), adds an iterative layer to the classical nGA and is designed to increase search efficiency by locating and converging on the low energy structures much faster than both nGA and PT.

1.1 Simulated annealing

Simulated annealing (SA) is a stochastic optimization method that can be used to find a good approximation of the optimal solution of a given objective function. Like most stochastic optimization methods, it is primarily warranted when the search space is too big (possibly even infinitely big) to be searched exhaustively, and it has been successfully applied to a number of NP-hard problems [6, 20, 15].

SA uses the analogy of the process that takes place when metals cool and freeze. Initially, when temperature is high, the system is disordered (melted metal), and atoms move around freely. When the temperature is then slowly lowered, atoms begin to arrange themselves in a way that minimizes the Gibbs free energy of the system. If the system is cooled slowly enough, the atoms will at the end have arranged themselves in a way that is believed to be at the Gibbs free energy minimum for that system. SA starts off with a (possibly random) start configuration, s, and then at each iteration of the algorithm it considers some neighbor, s', and moves to that state with a given probability, P. The probability depends on the current temperature, T, of the system and on the energy of the solution, E_s, usually such that

$$P(accept) = \begin{cases} exp^{-(E_{s'}-E_s)/T} & E_{s'} \geq E_s \\ 1 & E_{s'} < E_s \end{cases} \tag{1}$$

Equation 1 will ensure that high temperatures will increase the likelihood of a move to a state with a higher energy than the current state whereas low temperatures will decrease the likelihood of such a move. The temperature is set at each iteration according to a cooling scheme. The cooling scheme, i.e. how much the temperature is lowered at each step, needs to be chosen in a way that the algorithm will be allowed to search as much of the search space as possible while still allowing the algorithm to converge within the available time. It has been proven that if the cooling time is infinitely slow, the simulated annealing algorithm is guaranteed to find the optimal solution, but in practice there is, of course, little advantage of this theoretical proof as time is usually limited.

SA has proved to perform well in a number areas, such as lotto design [18], chip design [20] and of course protein structure prediction [2, 9, 12]. In fact, SA is without a doubt the most widely used algorithm for protein structure prediction [11]. One of the attractive features about SA for this particular problem is that because protein folding is also a molecular process, the terminology transcribes nicely. One might say that SA in a way mimics the actual folding process by continuously making small adjustment to the structure until it reaches its native state. Of course, SA is far too coarsely grained to actually simulate the folding process in practice - a molecular dynamics algorithm would be needed for that - but the approach still seem intuitively correct. In this respect, it should be noted that experiments have shown that molecular dynamics can in fact be used to successfully simulate the folding process and thereby predict protein structure [29]. Unfortunately, molecular dynamics is a very time consuming technique, and it is therefore currently not a serious option for predicting structures of any but the smallest of proteins.

Many of the best performing systems that compete in the CASP (Critical Assessment of protein Structure Prediction methods), including Rosetta, which has consistently won the competition for many years, are based on the SA-algorithm. However, it is very difficult to determine whether it is the simulated annealing approach that is responsible for the success achieved by the best-performing groups, or if it is in fact due to other circumstances such as choice of fragment library and energy function. A number of studies show that other algorithms outperform SA in various problems (see for instance [17]), but to my knowledge documentation of an actual benchmark between SA and any other algorithm for the protein structure prediction problem does not exist. We can merely conclude that it is a popular choice.

SA does not converge and is never completely trapped, but the algorithm may spend a long time escaping a local minima. Many researchers thus apply a reheating scheme that allow the SA process to escape local minima much faster. When the algorithm shows no improvement for a period of time, the temperature is simply elevated, thereby increasing the probability of the choosing a solution with a higher energy.

Because SA is a stochastic method, re-runs of the algorithm are typically performed. How many re-runs are required depends on the energy-landscape, but for the protein structure prediction problem re-runs are typically performed in the order of 300-400 times.

The pseudocode for a general SA algorithm is given in Algorithm 1.

Algorithm 1 Simulated Annealing

```
$temp \leftarrow temp_0$
$solution \leftarrow solution_0$
$energy \leftarrow evaluate(solution)$
for $i = 0$ to $n$ do
    $newSolution \leftarrow neighbour(solution)$
    $newEnergy \leftarrow evaluate(newSolution)$
    $deltaEnergy \leftarrow (newEnergy - energy)$
    if $random() < exp(-(deltaEnergy/temp))$ then
        $solution \leftarrow newSolution$
        $energy \leftarrow newEnergy$
    end if
    $temp \leftarrow adjustTemp()$
end for
return $solution$
```

1.2 Genetic algorithms

Genetic algorithms (GA) belong to the set of so-called population-based search algorithms. Unlike simulated annealing that operates on refining a single solution, population-based search algorithms operates on multiple solutions at the same time.

GA uses the analogy of genetic evolution to refine solutions by applying crossover and mutation to an initial population of (usually) random solutions. In each iteration of the algorithm, solutions are picked according to some selection criteria which usually depend heavily on the fitness of the solutions as calculated by some predefined fitness function. New solutions are then generated by pairing the selected solutions and combining parts of one solution with parts of the other solution (crossover) and altering the new solutions with a probability set by the mutation rate.

The GA does not allow the same degree of molecular process simulation as SA, and for the protein structure prediction problem, crossover definitely imposes a challenge because recombining part of one possible structure of a protein with a part of another possible structure of the protein easily results in steric clashes between atoms. This may also occur with SA, but in SA the changes are usually small, and the likelihood is thus less than with GA.

The advantage of GA over SA is that GAs tend to find good solutions faster because they cover a much larger and more diverse area of the search-space in a single run than do SA. However, because a GA jumps around on the energy-landscape it often makes only a very superficial scan of the different neighborhoods and may end up choosing to leave the neighborhood of the correct solution prematurely.

Genetic algorithms are also widely used to predict protein structures [26, 14], although not quite as popular as SA. The pseudocode for a standard genetic algorithm is given in Algorithm 2

Algorithm 2 Genetic Algorithm

```
$population \leftarrow createInitialPopulation()$
$evaluate(population)$
while $!done$ do
  $selectedPopulation \leftarrow select(population)$
  $population \leftarrow breed(selectedPopulation)$
  $evaluate(population)$
end while
```

1.3 Parallelism

With multi-core computers and clusters becoming more and more common, designing parallel algorithms that can utilize parallelism to improve both solutions quality and the speed with which we are able to obtain the solutions are in high demand. Both simulated annealing and genetic algorithms exist in parallel versions, and both outperform their sequential counterparts.

Due to the stochastic nature of meta-heuristics many re-runs are usually performed. Intuitively, we know that with N available processors we can perform N re-runs simultaneously, thereby gaining an overall speedup of factor N. However, it turns out that if the meta-heuristics are specifically designed to utilize parallelism, the solution quality actually exceeds that which can be expected from simply executing more re-runs [7].

Parallelism is an extra layer that can usually be added effortlessly to an existing sequential meta-heuristic. However, an optimal setting of the hyper-parameters for a sequential meta-heuristic is not necessarily the most optimal setting of the hyper-parameters for a parallel meta-heuristic. One of the key things for a successful meta-heuristic is to maintain a proper balance between exploration and exploitation. For simulated annealing, we traditionally use temperature to control this balance, and in genetic algorithms the selection strategy is responsible for maintaining the balance. In a parallel meta-heuristic, exploration is, however, largely maintained by running many parallel executions, which actually allow us to set the other hyper-parameters such that they favor exploitation to a much greater extent.

In the following two subsection we describe how the parallel versions of GA and SA work and propose a special variant of a parallel GA that is designed to specifically depend on parallelism to maintain the exploration-exploitation balance.

1.3.1 Parallel Tempering

To our knowledge, there exists only one truly parallel version of the SA algorithm known as the Parallel Tempering (PT) or Monte Carlo Replica Exchange algorithm [27, 7]. In PT, many simulations, or *replicas*, are started and run in parallel. The solutions are sampled in the same fashion as in the regular SA approach by making small alterations to the solution and accepting the change with a certain probability.

However, in a traditional SA process, the temperature is initially set high and then slowly lowered, but in PT each replica is run at a different but steady temperatures throughout the simulation. Replicas at lower temperatures favor exploitation whereas replicas running at higher temperatures favor exploration. To help replicas overcoming local minima barriers and to ensure proper exploration of the search space, replicas at temperatures i and j are periodically exchanged – traditionally with a probability of:

$$p = min(1, e^{(1/T_j - 1/T_i)(E_i - E_j)}) \tag{2}$$

where T_i and E_i are the temperature and energy of replica i respectively. In practice, this swapping scheme means that only structures at adjacent temperatures are swapped as the probability of replicas at distant temperatures approaches zero exponentially fast. A more aggressive swapping scheme may possibly improve the results of PT [4, 3].

PT (also know as replica exchange Monte Carlo search) is a parallel extension to the SA paradigm that has been shown to boost performance to an extent that exceeds the computational overhead associated with running multiple simulations [7]. It has been successfully applied to protein structure prediction previously [28, 30, 16, 22]. The pseudocode is given in Algorithm 3

1.4 Parallel genetic algorithm

Several parallel variants of the genetic algorithm exist and generally offer significant improvements by converging much faster at better solutions than the non-parallel version. There are two major approaches to parallelism in genetic algo-

Algorithm 3 Parallel Tempering

```
$initializeNreplicas$
$initializeNtemperatures$
$bestSolution$
while !$done$ do
    for $i = 0$ to $N$ do
        $ExecuteSimulated_Annealing(replica_i, steps)$
        if $Energy(replica_i) < Energy(bestSolution)$ then
            $bestSolution \leftarrow replica_i$
        end if
    end for
    for $i = 0$ to $N - 1$ do
        for $j = i + 1$ to $N$ do
            if $random() < exp(f(deltaTemp) * deltaEnergy)$ then
                $Swap(i, j)$
            end if
        end for
    end for
end while
$returnbestSolution$
```

rithms. One is often referred to as the master-slave model, where a single process (the master) controls the genetic algorithm, but uses a number of other processes (the slaves) to evaluate and possible breed the individuals. The slave processes are run in parallel. Compared to the sequential GA, this parallelization scheme will only have an impact on speed – not on solution quality.

Another parallelization paradigm that will generally also have a positive effect on solution quality is the niche model (also known as the island hopping or deme model). A niche genetic algorithm (nGA) is a parallel implementation of a genetic algorithm where sub-populations evolve independently from each other on different islands, which can be run in different threads or on different processors. At certain points during evolution, individuals migrate from one niche to another and become part of the population of that niche. nGA exploits the fact that different runs of the same genetic algorithm are likely to produce different suboptimal solutions that combined are likely to yield better results. The convergence rate of a nGA is strongly affected by the migration scheme [1]. Migrating and replacing only a few randomly chosen individuals leads to very slow convergence whereas migrating the best and replacing the worst leads to the fastest convergence. Pseudocode of a traditional nGA is given in Algorithm 4

As a special version of nGA, we propose an iterative niche genetic algorithm

Algorithm 4 nGA

```
$InitializeNniches$
$X \leftarrow NumberOfIndividualstomigrate$
$bestSolution$
while !$done$ do
  for $n = 0$ to $N$ do
    $ExecuteGenetic_Algorithm(n, steps)$
    if $Energy(niche_n) < Energy(bestSolution)$ then
      $bestSolution \leftarrow niche_n$
    end if
  end for
  for $n = 0$ to $N$ do
    $migrants \leftarrow CloneIndividuals(n, X)$
    $RemoveFromPopulation(X)$
    $Migrate(migrants, n + 1)$
  end for
end while
$returnbestSolution$
```

(inGA). Instead of simply migrating m individuals between niches at different time intervals like the nGA, inGA evolves populations for a certain time t, then picks the best individual from each niche, makes n copies and place one copy on n new niches. The old niches are discarded completely. The algorithm is iterated until convergence or until a certain criterion is met. The pseudocode for the algorithm is shown in Algorithm 5

Essentially, the strategy corresponds to letting all niches converge before migrating individuals between them and restarting as described by Cantu-Paz and Goldberg [5]. However, while running each niche to convergence worked well for the problem instances chosen by Cantu-Paz and Goldberg, work by Heiler [10] suggests that for protein structure prediction the quality of predicted structures *decreases* when the individuals are locally optimized before the genetic operators are applied. This is actually somewhat surprising and seems to oddly go against the general idea of genetic algorithms, but we recognize that the biggest improvements happen during the first generations, and rather than running to convergence we thus suggest a kind of early stopping. This way, we generate low energy structures without spending too much time on refining suboptimal structures.

The appeal of inGA is that it actually *depends* on parallelism rather than time to improve solution quality. In order to find good solutions in highly rugged fitness landscapes, the exploration-exploitation must be maintained. For a standard GA the exploration-exploitation balance is usually maintained by the selection strategy alone. For nGA the migration scheme and the number of niches also exert control over this balance although, it does not depend on parallelism to maintain it. The nature of inGA is, however, highly elitist, and it is thus crucial to include a large number of niches as exploitation would otherwise be far too heavily favored. Fortunately, CPU power continues to steadily improve and computer clusters are more and more common, and making solution quality depend on parallelism rather than on time is thus a very attractive feature indeed.

Algorithm 5 inGA

```
$InitializeNniches$
$bestSolution$
$bestSolutions \leftarrow CreateRandomInitialSolutions()$
while $!done$ do
  $population \leftarrow Clone(bestSolutions)$
  for $n = 0$ to $N$ do
    $ExecuteGenetic_Algorithm(population, n, steps)$
    if $Energy(niche_n) < Energy(bestSolution)$ then
      $bestSolution \leftarrow niche_n$
    end if
  end for
  for $n = 0$ to $N$ do
    $bestSolutions[n] \leftarrow CloneBestSolution(n)$
  end for
end while
$returnbestSolution$
```

2 Experiments

Experiments with the different meta-heuristics have been carried out. In order to easily compare the performance, the same energy functions and test proteins are used in all experiments. The same move set is also constructed, such that the accessible part of the conformational space is the same in all experiments. The meta-heuristic settings are described in detail in the following subsections.

2.1 Encoding

A physics-based energy potentials, called POISE [19], which requires a full atom model is used as the objective function for the search algorithms. We thus need to encode the protein either with every atom and its position explicitly represented or in a way that allows us to calculate the position of all atoms. The latter is usually preferred over the former, because it is much easier to enforce restrictions in the space of dihedral angles, bond angles and bond lengths than in the Cartesian space. In the following, we refer to the dihedral angles, bond angles and bond lengths of an amino acid as the set of structural variables, S. Note that given S the exact positions of all atoms for that amino acid can be calculated directly by using standard matrix operations. An entire protein is thus encoded as a vector of S, where S_i represents the structural variables of the amino acid at position i.

One of the problems often encountered during encoding of proteins is the occurrence of clashing atoms. In real life, this will never happen, as the energy of such a protein would rapidly grow towards infinity and the atoms would be forced apart. However, in a simulation one of two strategies can be utilized: one can either explicitly check and make sure that atoms do not clash and simply discard solutions where clashes do occur, or clashes can be tolerated but heavily penalized by the energy function, such that solutions with clashing atoms stand little chance of being accepted/selected. The latter works best in conjunction with statistics-based energy function where the "energy" term is usually an artificial pseudo energy made up from many parameters that are non-numerical in nature. With a pure physics-based energy potential, the infinitely large increase in energy caused by two clashing atoms will cause overflow on a computer which, needless to say, is highly unfortunate. As we utilize a pure physics-based potential, a check for clashing atoms is thus carried out before a solution is evaluated, and only clash-free structures are accepted.

The protein is encoded sequentially, and for every new residue added we check whether the S_i chosen for residue i causes any of the atoms of the new amino acid to clash with atoms in the residues that have already been added. If a clash occurs a new S_i for the amino acid is chosen. If the problem with atom clashes has not been resolved after 20 different S_i have been tried, we backtrack and choose a new S_{i-1} for the previous residue. Although the theoretical running time for this strategy is $O(2^n)$, the running time was not found to be an issue in practice.

In order to avoid spending time on checking for clashes between atoms that are too far away in space, the space is divided into cubes. The cubes can be of arbitrary size, but as we want to limit the amount of calculations we need to perform, they should be kept quite small. We have chosen a size of 2x2x2 Å. We use a hash map with buckets to store and associate cubes with atoms to ensure fast retrieval of atoms. Every time the coordinates of an atom is calculated, the corresponding cube is determined, and the atom is then placed in the hash map entry corresponding to that cube. A check it performed to ensure that the atom does not clash with any of the atoms already in the same or in an adjacent cube.

2.2 Move set

The move set is defined as the set of possible combinations of bond lengths, angles and dihedral angles for each amino acid. Theoretically, the move set is unrestricted, but in practice we know that bond lengths and angles vary very little and dihedral angles are heavily biased towards certain areas of the dihedral angle space. Here, we thus choose bond angles and bond lengths for amino acids randomly from within a small interval (up to ±0.1 Å) of the optimal angles and lengths as defined in the AMBER 99 parameters.

The dihedral angles space is likewise restricted. In Helles and Fonseca [8] a probability distribution is predicted for each amino acid in a sequence by considering the neighboring amino acids. This probability distribution is used here such that the dihedral angles are chosen from this sequence-dependent distribution, thereby maximizing the probability of sampling a realistic area of the dihedral angle space.

2.3 Energy function

It is well known that the success of search algorithms is correlated with the energy function used [25], and it is thus important to use an energy function that is representative for the problem at hand - in this case protein structure prediction. Generally speaking, energy functions in this field of research can be divided into two categories: physics-based energy function and statistics-based energy functions. The energy function used here, called POISE, is a purely physics-based potential, described in more details in [19]. It combines the AMBER force field [27]:

$$\begin{aligned} E_{\text{protein}} = &\sum_{\mathrm{i}}^{bonds} K_b(b_i - b_0)^2 + \sum_{\mathrm{i}}^{angles} K_\theta(\theta_i - \theta_0)^2 \\ &+ \sum_{\mathrm{i}}^{dihedrals} k_x[1 + \cos(n\phi - \gamma)] + \\ &+ \sum_i^N \sum_{i<j}^N \left(\frac{q_i q_j}{r_{ij}} + 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] \right) \end{aligned} \tag{3}$$

with a Generalized Born component:

$$V_{\text{GB}} = -\frac{1}{2} \left(\frac{1}{\epsilon_p} - \frac{1}{\epsilon_w} \right) \sum_i \sum_{i<j} \frac{q_i q_j}{f_{ij}^{GB}(r_{ij})} \tag{4}$$

$$f_{\mathrm{ij}}^{GB}(r_{ij}) = \left[r_{ij}^2 + R_i R_j exp\left(\frac{-r_{ij(2)}}{4R_i R_j} \right) \right]^{\frac{1}{2}}$$

and a hydrophobic mean force potential:

$$V_{\mathrm{HMFP}} = \sum_{i \in SA_i > A_c}^{N_c} tanh(SA_i) \sum_{j \in SA_j > A_c, j \neq i}^{N_c} tanh(SA_j)$$
$$x \sum_{k=1}^{3} h_k exp\left(-\left[\frac{r_{ij} - c_k}{w_k} \right]^2 \right) \quad (5)$$

We refer to [19] for an in-depth explanation of the parameters.

The potential considers interatomic energies between all pairs of atoms, and the time complexity of calculating the potential is thus quadratic ($O(n^2)$). Fortunately, the vast majority of atoms in a protein are simply too far apart to exert any power over each other and by simply omitting calculations of these interactions, the function is implemented such that the potential is calculated in linear time.

It should be noted that in this current work we are primarily concerned with exploring parallel meta-heuristics for the protein structure prediction problem. We have chosen the POISE energy function, because it gives a realistic impression of the very rugged fitness landscape that the algorithms have to navigate around in. We do not have an opinion about which energy function is better. Indeed, another energy function may direct the search in a very different direction than the POISE potential, but for evaluating the performance of meta-heuristics this is of little interest. We are only concerned with minimizing energy according to some realistic energy function.

2.4 Test set

In lack of a standardized, diverse set of proteins that is guaranteed to provide an adequate representation of protein structures, we have constructed a test set that includes 2 proteins from each of the categories α, β and $\alpha\beta$ in the PDB Select 25% [2] data set. Only small to medium sized proteins have been included. The smallest protein includes 46 amino acids and the largest protein includes 81 amino acids (Table 1).

2.5 Sequential algorithms

Although the parallel algorithms are expected to perform better, we have included the sequential algorithms in the experiment as a baseline for performance.

[2]http://bioinfo.tg.fh-giessen.de/pdbselect/recent.pdb_select25

	Category	Residues
1C75	α	71
1NKD	α	59
1YK4	β	52
2O9S	β	67
1EJG	$\alpha\beta$	46
1IQZ	$\alpha\beta$	81

Table 1: Test proteins

2.5.1 Simulated annealing

The simulated annealing procedure starts at a high temperature where solutions of higher energy still have a fair chance of being selected and then cools down accepting fewer and fewer high-energy solutions. What exactly constitutes a "high" energy depends on the differences between energies of different structures. If the average difference in energy between two solutions is large then "high" also translates into a large number, like say 10.000 or more. If, on the other hand, the energy difference on average is small, then "high" may translate into a number that we would normally consider low, like 100 or less. Start and final temperatures are here determined in the way proposed in [24], such that

$$T_{start} = -\delta E_{max}/ln(P^A(\delta E_{max})) \tag{6}$$

and

$$T_{final} = -\delta E_{min}/ln(P^A(\delta E_{min})) \tag{7}$$

Initial experiments measuring differences in the energy, E, between neighboring structures were run to determine the values of E_{max} and E_{min}. $P^A(\delta E_{max})$ and $P^A(\delta E_{min})$ were set to 0.95 and 0.05 respectively. This resulted in $T_{start} = 3800$ and $T_{final} = 10$.

In each time step, we attempt N moves for a N-residue long protein. A move consists of randomly selecting an amino acid and picking a new set of structural variables, S_i, for that amino acid. We use the standard probability of accepting a neighboring structure as described in section 1.1.

The standard exponential cooling scheme is applied

$$T_{k+1} = \alpha * T_k \tag{8}$$

where α is the cooling rate. α was set at 0.99 resulting in $\approx$1000 steps to cool down the system. Once the system has been cooled down and shows signs of convergence (i.e. the solution has not improved for a number of iterations), we re-heat the system, thereby allowing it to escape local minima faster.

2.5.2 Genetic algorithm

We used a population size of 20 individuals. All individuals are encoded as described above and evaluated using the POISE energy potential. Before evaluation the exact positions of all atoms are calculated from the set of structural variables, S, using standard matrix operations.

The selection strategy of the GA employes both an elitism strategy and the fitness proportionate selection strategy known as a roulette wheel. The elitism strategy copies and transfers the 10% top scoring individuals unaltered to the next generation, thereby ensuring that the best individuals are always kept. However, the 10% best individuals are also allowed to compete in the roulette wheel selection where each individual is chosen with a probability corresponding to its fitness value. This strategy is chosen over rank selection to ensure a better chance for low scoring individuals to be selected.

Individuals selected by the roulette wheel are subjected to crossover and mutation. A multi-point crossover strategy is used where the number of crossover sites, n, are chosen according to a Gaussian distribution and the n actual sites chosen at random. The advantage of multi-point crossover over single point crossover is that it eliminates the bias of the end segments that is commonly raised as an issue with the vector representation employed by most genetic algorithms. Additionally, multi-point crossover typically results in bigger alterations of the solutions, causing the genetic algorithm to explore very different regions of the search space.

As is often the choice in genetic algorithms, the mutation rate is set fairly low to a value of 0.001. Mutation is thus not the driving force in the folding process, but is used mainly as a way to introduce new genes into the existing gene pool.

The crossover and mutation operations frequently result in clashing atoms. We solve this in much the same way as during encoding. Thus, when the atoms of an amino acid clash with another, we allow 20 attempts of local adjustments (i.e. picking a new set of structural variable) for that amino acid. If it continues to clash, we backtrack and allow 20 attempts of local adjustments of the previous amino acid. The clashing problem is nearly always resolved within a few backtracks, although it may occasionally require more than 10 backtrack attempts.

2.6 Parallel algorithms

In order to make a fair comparison of the different approaches, the parallel algorithms are constructed such that the hyper-parameters are the same as for the sequential algorithms where possible. Of course, some alterations to the GA and

SA are needed, because parallelism adds an extra layer of complexity that requires minor adjustments.

2.6.1 Parallel Tempering

The PT algorithm is implemented such that it utilizes the same encoding strategy, energy function, move set and probability of accepting a move as described above. The lowest temperatures is equivalent to the one stated in the sequential SA approach (10). The highest temperature is slightly higher at 4200 (see Equation 9). PT does not employ a cooling scheme but instead runs replicas at different temperatures within the interval of the lowest and highest temperature. Temperatures can be either spaced in any arbitrary way, but Kone and coworkers suggested that the observed average probability of accepting a swapping move between neighboring replicas should be roughly 20% [13]. Temperatures for our experiments have been spaced according to:

$$T_{replica_{i+1}} = 20i + T_{replica_i} \tag{9}$$

This results in lower temperatures being closer than higher temperatures. We have chosen this to increase the likelihood of swapping replicas running at lower temperatures, because low temperature replicas otherwise stand little chance of escaping local minima. Replicas running at higher temperatures will always have a higher probability of choosing a higher energy structure and will thus not easily be trapped.

The probability of accepting a swap between to replicas, i and j, was given by:

$$P(i \leftrightarrow j) = min\{1, exp^{[(\beta_i - \beta_j)(E_i - E_j)]}\} \tag{10}$$

where β is the inverse temperature $\beta = 1/T$ and $\beta_i > \beta_j$. Defining the probability of swapping replicas such that it decreases exponentially as the gap between temperature increase is usually employed in PT [7] and also the reason why it was chosen here.

We ran 20 parallel simulations which ensured proper communication between replicas, e.i. swapping did occur between adjacent replicas as expected.

2.6.2 Niche genetic algorithms

We made experiments with two different niching algorithms: the traditional nGA and the proposed iterative nGA (inGA). For both parallel algorithms we used the sequential GA described above to evolve all sub-populations, except we exchanged the combination of roulette wheel and elitist selection with a pure elitist selection strategy. In order to also make them comparable with the PT algorithm, we ran 20 parallel niches for both nGA and inGA.

For nGA we settled on a rather strong migration scheme such that every 100 generation we chose the 50% best individuals from each population, cloned them and migrated them to another niche where they replaced the 50% worst individuals. We chose this migration scheme over a softer migration scheme to ensure faster convergence and make it more comparable with inGA.

inGA evolves each niche for 100 generations and then chooses the best from each niche, clones them, distributes them and continues evolution.

We chose 100 generations for the niche algorithms, because we observed the by far largest improvement during the first 100 generations. As the best solution from each niche was guaranteed to proceed into the next iteration, allowing further refinement of that particular solution, it thus seemed prudent to stop at this point. We did run a few experiments with more generations, but found no improvement in final solution quality and therefore settled on 100 generations.

3 Results and discussion

Results from experimentation with the three different parallelization schemes on the test proteins are shown in Table 2. We made 200 restarts of the sequential algorithms, SA and GA, and 20 restarts of the parallel algorithms, PT, nGA and inGA. The algorithms were executed on a HP ProLiant DL360 G6 server with 2 Xeon X5550 2.66 GHz quad core processors, and each algorithm was allowed a maximum of 72 hours of CPU time. We report the average result.

From the results it is evident that inGA is quickly and consistently locating structures of lower energy than both nGA and PT. Please note that we have not calculated RMSD of the final structures, because as such the search algorithms are all oblivious to the concept of a native structures. They seek merely to minimize energy as specified by the POISE potential, and in this experiment we are only interested in determining how efficient the different algorithms are in finding low energy structures in the highly rugged energy landscape associated with protein structure prediction energy functions. A different energy function would most likely lead to different (either better or worse) quality of the final structures in terms of RMSD to the native structure, but the differences in how well the algorithms perform with respect to each other would (expectedly) remain the same.

Given unlimited time, all meta-heuristics would probably find the same low energy structures. Unfortunately, time is usually not unlimited in practice, and designing search algorithms that increase search efficiency such that we can obtain better results faster becomes important. Parallelization has in itself increased search efficiency, but from the results presented here it is evident that how the algorithms are parallelized can also have a profound impact on how efficiently the

	nGA	inGA	PT	GA	SA
1C75	368.0	**267.4**	476.1	938.5	1219.8
1NKD	340.0	**13.6**	267.3	613.9	1345.1
1YK4	212.8	**148.7**	239.8	589.6	902.4
2O9S	78.6	**0.4**	191.3	312.0	538.0
1EJG	42.7	**-23.8**	8.0	270.1	299.1
1IQZ	589.7	**359.49**	911.6	1967.3	2345.7

Table 2: Average results obtained with the different algorithms after 72 hours of execution time on 8 CPUs (see text). Energies are calculated with the POISE potential. Lower energies are better

algorithms travel the energy surface in their search for the global minima.

The solution space for a given protein sequence is infinitely big and the key to success for a meta-heuristics is usually a good balance between exploration and exploitation. Minima should be explored thoroughly while still allowing the algorithm to move relatively freely across energy barriers. PT, nGA and inGA all differ from each other in this exploration-exploitation balance.

In PT the balance between exploration and exploitation is kept by running parallel simulations at different temperatures. The advantage of PT is that it can be run for exactly as long as time permits, because while it may settle at a minima, it does not really converge but rather keeps exploring for a preset number of iterations or until it is interrupted. As such, the PT algorithm enjoys the same theoretical guarantee of finding the global minima as the simulated annealing algorithm. However, while parallel tempering reaches low energy structures faster than sequential Monte Carlo simulations [7], the number of replicas used depends not so much on available processors, but on what makes sense in order to maintain proper communication between the different replicas. In other words, there appears to be an upper limit to what we can expect to gain in performance that depends on the problem and not on CPU power. For proteins of the length used here, 20 replicas ensure appropriate communication across temperatures, and more replicas would thus only increase the level of communication, thereby setting off the exploration-exploitation balance which would not be desirable. Of course, for larger proteins where the energy span between different structures is likely to be greater than for the proteins used here, more replicas would most likely be required to ensure proper communication.

One of the reasons why PT does not reach the low energy structures as fast as the genetic algorithms is that, although many replicas are run at the same time, they do not exchange information between replicas. If a good solution is encountered at one temperature, it may be exploited by swapping it to a lower temperature, but it does not share its favorable characteristics with any of the other replicas. Parallel tempering would most likely reach the same results as achieved by inGA, but we postulate that because of the lack of information sharing it can generally be assumed to take much longer.

The genetic algorithms, on the other hand, have a high degree of information sharing via their crossover operator, which explains why the genetic algorithms reach the lower energy structures much quicker. The migration scheme we have used here for nGA is fairly aggressive to ensure faster convergence that would be comparable with inGA. From the results it is evident that while nGA finds lower energy structures than PT, it does not reach structures with energies as low as inGA. We did initially experiment with a less aggressive migration scheme (that migrated only the best individual), but energies were significantly worse after the 20 iterations than with the chosen migration scheme.

An issue with inGA is that it may simply converge prematurely. The iterative strategy of inGA is highly elitist, and with 20 niches it does usually converge or show sign of convergence within 20 iterations for the small to medium sized proteins used here. An elitist strategy (always picking the best) favors exploitation heavily and will normally only work well in smooth energy landscapes. The energy landscape of proteins is obviously anything but smooth, but interestingly a balance with exploration does nevertheless appear to be maintained in inGA by the niche approach. Exploration can thus be controlled by simply adding more or less niches. This is indeed a nice feature, since it allows us to focus all our computational power on finding better local energy minima – without having to worry about escape options – which in turn leads us faster towards structures of lower energy. Obviously, adding more niches would most likely require more iterations, but the number of iterations required to find the low energy structures would expectedly grow much slower for inGA than for nGA thereby making the difference in performance between inGA and nGA greater as the number of niches increase.

4 Conclusion

PT and nGA, with N replicas and niches respectively, essentially require N times more computational time than a single run of their sequential counterparts. However, with multi-core computers and CPU clusters being readily available to most researchers, they can be executed in parallel, and the extra computational time required does not impose a problem. The experiments in this report confirm that PT and nGA search more efficiently and arrive at much better results than their sequential counterparts. They further demonstrate that the actual parallelization scheme have a significant impact on search efficiency with the proposed algorithm, inGA, clearly outperforming both the PT algorithm and the traditional nGA.

References

[1] E. Alba. *Parallel Metaheuristics.* Wiley, 2005.

[2] P. Bradley, K. M. S. Misura, and D. Baker. Towards High-Resolution de Novo Structure Prediction for Small Proteins. *Science*, 309:1868–1871, 2005.

[3] P. Brenner, C. R. Sweet, D. VonHandorf, and J. A. Izaguirre. Accelerating the replica exchange method through an efficient all-pairs exchange. *J. Chem. Phys.*, 126:074103+, 2007.

[4] F. Calvo. All-exchanges parallel tempering. *J. Chem. Phys*, 123:124106+, 2005.

[5] E. Cant-Paz and D. E. Goldberg. Modeling idealized bounding cases of parallel genetic algorithms. In *In*, pages 353–361. Morgan Kaufmann Publishers, 1996.

[6] W. Chiang and R. A. Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *A. Oper. res.*, 63:3–27, 1996.

[7] D. J. Earlab and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem*, 7:3910, 2005.

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

[9] Y. Fujitsuka, G. Chikenji, and S. Takada. SimFold Energy Function for De Novo Protein Structure Prediction: Consensus with Rosetta. *Proteins*, 62:381–398, 2006.

[10] M. Heiler. Massively parallel gas for protein structure, 1998.

[11] G. Helles. A comparative study of the reported performance of *Ab Initio* protein structure prediction algorithms. *J. R. Soc. Interface*, 5:387396, 2008.

[12] L. Hung, S. Ngan, T. Liu, and R. Samudrala. PROTINFO: new algorithms for enhanced protein structure predictions. *Nuc. Acids Res.*, 33:Online, 2005.

[13] A. Kone and D. A. Kofke. Selection of temperature intervals for parallel-tempering simulations. *J. Chem. Phys.*, 122:206101, 2005.

[14] F. Koskowski and B. Hartke. Towards Protein Folding with Evolutionary Techniques. *J. Comp. Chem*, 26:1169–1179, 2004.

[15] P. J. M. Van Laarhoven, E. H. K. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operation research*, 40:113–125, 1992.

[16] D. Latek, D. Ekonomiuk, and A. Kolinski. Protein Structure Prediction: Combining De Novo Modeling with Sparse Experimental Data. *Wiley InterScience*, online, 2007.

[17] T. W. Leung, C. H. Yung, and M. D. Troutt. Applications of genetic search and simulated annealingfo the two-dimensional non-guillotine cutting stock problem. *Comp. Ind. Eng.*, 40:201–124, 2001.

[18] P.C. Li and G.H.J Van Rees. Lotto design tables. *J. Comb. Designs*, 10:335–359, 2002.

[19] M. S. Lin, N. Lux Fawzi, and T. Head-Gordon. Hydrophobic potential of mean force as a solvation function for protein structure prediction. *Structure*, 15:727–740, 2007.

[20] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. Comp. Des. Int. Circ. Sys.*, 15:1518–1524, 1996.

[21] M. T. Oakley, D. Barthel, Y. Bykov, J. M. Garibaldi, E. K. Burke, N. Krasnogor, and J. D. Hirst. Search strategies in structural bioinformatics. *Curr. Prot.Pep. Sci.*, 9:260274, 2008.

[22] S. B. Ozkan, G. A. Wu, J. D. Chodera, and K. A. Dill. Protein folding by zipping and assembly. *PNAS*, 104:11987–11992, 2007.

[23] G. N. Ramachandran and V. Sasisekharan. Conformations of polypeptides and proteins. *Adv. Prot. Chem.*, 23:283–437, 1968.

[24] H. Sanvicente-Snchez and J. Frausto-Sols. A method to establish the cooling scheme in simulated annealing like algorithms. *LNCS*, 3945:755–763, 2004.

[25] A. Schug, T. Herges, A. Verma, and W. Wenzel. Investigation of the parallel tempering method for protein folding. *J. Phys.*, 17:S1641–S1650, 2005.

[26] A. Schug and W. Wenzel. An Evolutionary Strategy for All-Atom Folding of the 60-Amino-Acid Bacterial Ribosomal Protein L20. *Biophys. Jour.*, 90:4273–4280, 2006.

[27] Robert H. Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Phys. rev. let.*, 57:2607–2609, 1986.

[28] J. S. Yang, W. W. Chen, J. Skolnick, and E. I. Shakhnovich. All-Atom Ab Initio Folding of a Diverse Set of Proteins. *Structure*, 15:53–63, 2006.

[29] B. Zagrovic, C. D. Snow, M. R. Shirts, and V.S.Pande. Simulation of Folding of a Small Alpha-helical Protein in Atomistic Detail using Worldwide-distributed Computing. *J. Mol. Biol.*, 323:927–937, 2002.

[30] Y. Zhang, A. Kolinski, and J. Skolnick. TOUCHSTONE II: A New Approach to Ab Initio Protein Structure Prediction. *Biophys. Jour*, 85:1145–1164, 2003.

Chapter 4

Restricting the PSP search space

4.1 Background

The probability that a search algorithms will uncover the global optimal solution to a given problem depends highly on the ruggedness of the search landscape. Figures 4.1a and 4.1b show a smooth and a rugged energy landscape respectively. Clearly, it is much easier to locate the global minimum in a smooth landscape than in a rugged landscape. In fact, a smooth landscape seem to guide us directly towards the global optimum.

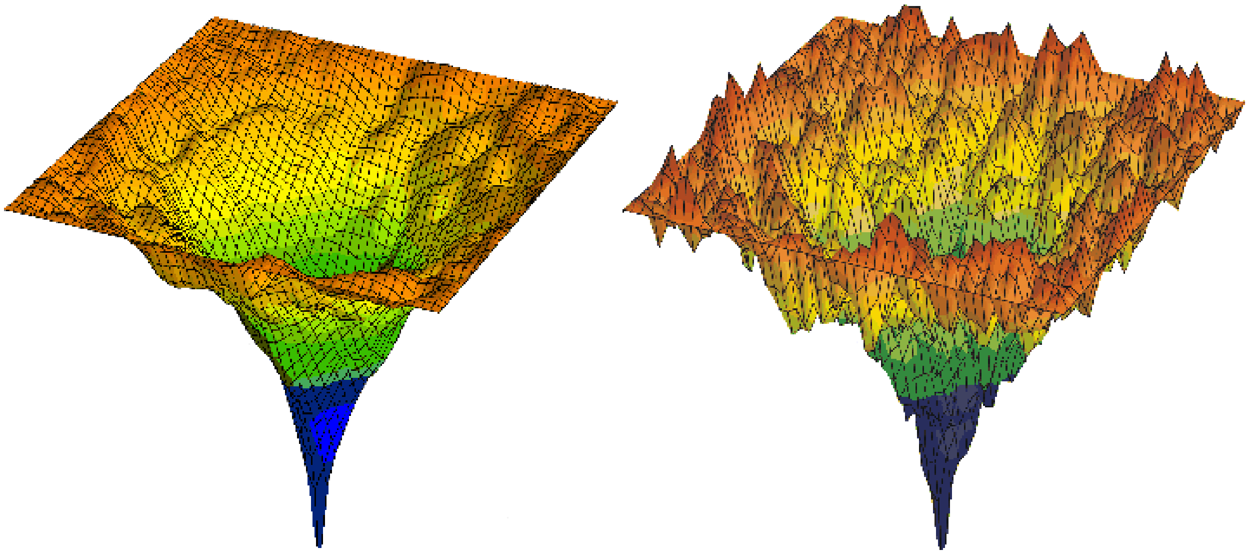

(a) Smooth landscape (b) Rugged landscape

Figure 4.1: Energy landscapes

Unfortunately, the energy landscape of proteins tends to be extremely rugged. Especially physics-based energy potentials are difficult, because even the slightest change in structure can cause huge differences in energy, creating a landscape that is more or less impossible to navigate in. Statistics-based energy potentials generally result in smoother (but still rugged) energy landscapes, but of course suffer from being more imprecise than the physics-based potentials.

Since a perfect energy function that would direct a search algorithm straight to the global optimum is thus out of our (current) reach, we are forced to explore other paths. A particular appealing approach is to try to narrow down the search space. We may, for instance, choose to discard the part of the search space that contains structures where two non-covalently bound atoms get within their van der Waals radii, because we know that the energy grows exponentially fast in those cases, and the structure will thus not be at a energy minimum. We may even take it a little further and choose to cut away parts of the search space simply because we find it unlikely to contain the native structure. Or we could look at it the other way around and instead choose to concentrate our search in areas that contain certain substructures that we deem highly probably. PSP algorithms that use fragment assembly or secondary structure predictions are examples of this.

The following two papers pertain to restrictions of the search space for PSP.

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

This paper has been published in BMC Bioinformatics in 2009 [4]. It is joint work with Rasmus Fonseca.

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

Glennie Helles   Rasmus Fonseca

Abstract

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.

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 $30^\circ \times 30^\circ$ 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 ($\approx 80\%$) is achieved by observing the 20 bins with highest output values.

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 [2] 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 [1,21,25]. Fortunately, predicting secondary structures can be done quite accurately [4, 11, 20, 23], and since roughly 60% of amino acids in most proteins are arranged in these secondary structures [5], 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. [15] 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. [13] 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 [16, 18, 26]. A significant amount of work has already been done in predicting these local fragments [7–10, 14, 24], but as noted in [8], 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 [11, 20, 23].

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) [17]. 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 [27]. 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 [3], 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.

Methods

A fully connected feed-forward neural network was constructed and used to predict a $30^\circ \times 30^\circ$ 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 [12]. 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 [6]. 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 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 [20] to predict secondary structure. Figure 1 shows an overview of the neural network design.

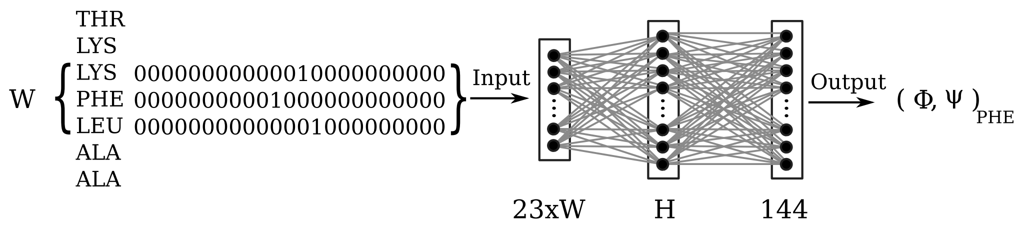

THR
LYS
W
LYS 00000000000010000000000
PHE 00000000001000000000000
LEU 00000000000001000000000
ALA
ALA
Input
Output
(Φ, Ψ)PHE
23xW
H
144

Figure 1: Network configuration. 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 144 neurons in the output-layer each correspond to a $30° \times 30°$ 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 over-fitting.

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 previ-

	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 %

Table 1: 5-fold cross validation results. 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.

ously 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 1, the neural network was able to predict the correct $30° \times 30°$ bin approximately 16% of the times regardless of the way the data set was split.

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 2 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:

$$G = \frac{R_{\text{most}}}{R_{\text{total}}} \tag{1}$$

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 4 B 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.

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%

Table 2: Improvements in prediction accuracy. 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

As can be seen from Table 2 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 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 compared to baseline statistics is achieved when we look at the 8 bins with highest output values.

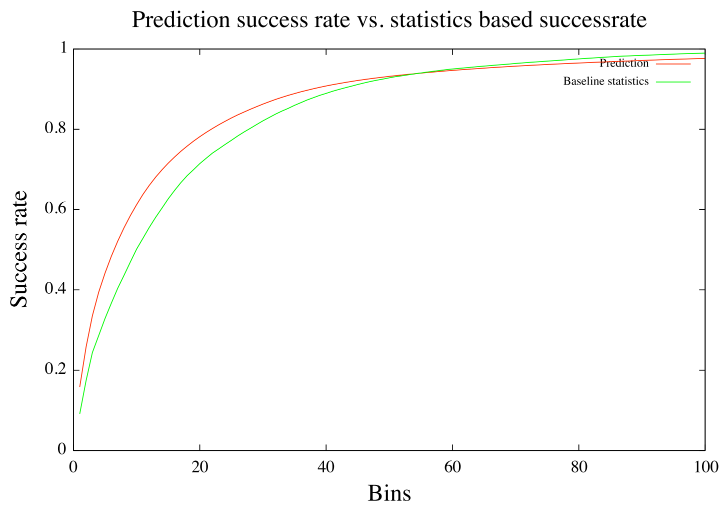


Figure 2: NN prediction vs. baseline statistics

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 $\approx 80\%$ 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.

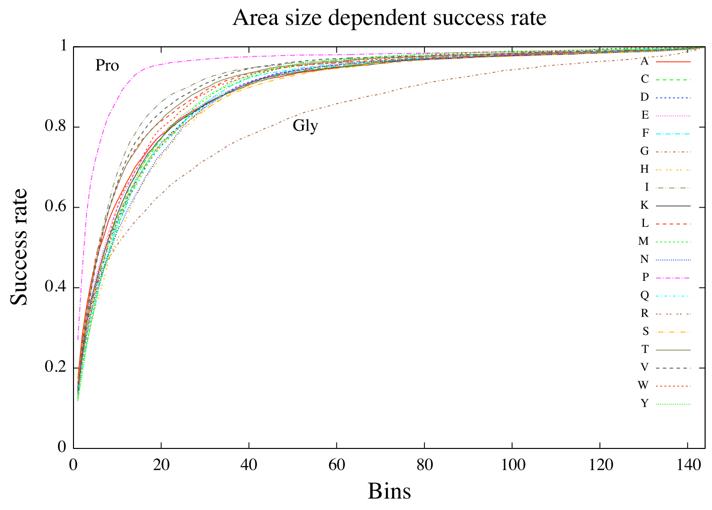


Figure 3: Area size dependent success rate. Each bin represents a 30° × 30° area of the Ramachandran plot.

Comparison

Both Kuang et al. [17] and Zimmermann & Hansmann [27], who attempted to predict dihedral angle areas of coil residues, divided the Ramachandran plot into three areas. Their smallest area (area A in [17], area H in [27]) has roughly the same size as 21 of our 30° × 30° bins. The second smallest area (area B in [17], area E in [27]) has an area corresponding to 25 of our bins and the largest area (referred to as area E/G in [17] and area O in [27]) corresponds to more than 80 of our bins - in fact in [27] 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 [17] and us. Generally, sequences with 50% or more sequence identity can be assumed to adopt the same three-dimensional structure whereas structures with only 25% cannot [22]. 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 [17, 27] 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 4 C shows an example of the area predicted for threonine in a randomly chosen sequence from the data set. Figure 4 A and B 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.

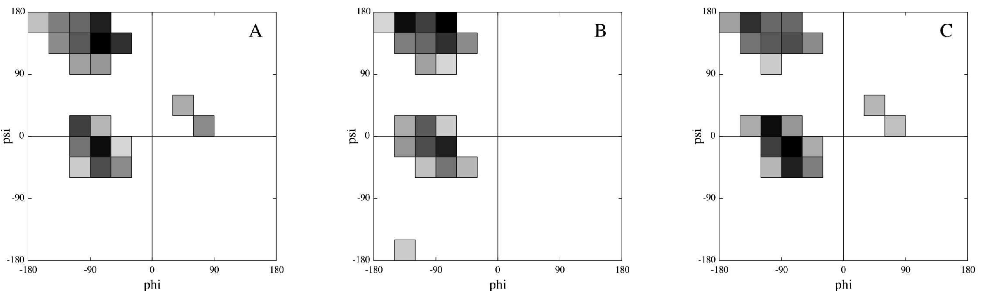


Figure 4: Bin distribution. The plot to the left (A) shows the distribution of the 20 most populated 30° × 30° bins for all coil residues in the training set. The plot in the middle (B) shows the distribution for just threonines in the training set, and the plot to the right (C) shows the predicted bins 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.

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 [19] 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.

Conclusions

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 $\approx$ 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.

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] D. J. Barlow and J. M. Thornton. Helix geometry in proteins. *J. Mol. Biol.*, 201:601–19, 1988.

[2] M. Ben-David, O. Noivirt-Brik, A. Paz, J. Prilusky, J. L. Sussman, and Y. Levy. Assessment of casp8 structure predictions for template free targets. *Proteins*, epub:00–00, 2009.

[3] W. Boomsma, K. V. Mardia, C. C. Taylor, J. Ferkinghoff-Borg, A. Krogh, and T. Hamelryck. A generative, probabilistic model of local protein structure. *PNAS*, 105:8932–8937, 2008.

[4] W. Chu, Z. Ghahramani, and D. L. Wild. A graphical model for protein secondary structure prediction. In *Proc 21st Int. Conf. Mach. Learn.*, page 21, 2004.

[5] T. Creighton. *Proteins - Structures and molecular properties*. Freeman, 1993.

[6] J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, 34:508–19, 1999.

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

[8] N. Fernandez-Fuentes, B. Oliva, and A. Fiser. A supersecondary structure library and search algorithm for modeling loops in protein structures. *Nucl. Acids Res.*, 34(7):2085–2097, 2006.

[9] L. Fourrier, C. Benros, and A. G. de Brevern. Use of a structural alphabet for analysis of short loops connecting repetitive structures. *BMC Bioinformatics*, 5:58, 2004.

[10] C. G. Hunter and S. Subramaniam. Protein local structure prediction from sequence. *Proteins*, 50:572–579, 2003.

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

[12] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[13] W. Kabsch and C. Sander. On the use of sequence homologies to predict protein structure: Identical pentapeptides can have completely different conformations. *PNAS*, 81:1075–1078, 1984.

[14] S. Katzman, C. Barrett, G. Thiltgen, R. Karchin, and K. Karplus. Predict-2nd: a tool for generalized protein local structure prediction. *Bioinf.*, 24:2453–2459, 2008.

[15] O. Keskin, D. Yuret, A. Gursoy, M. Turkay, and B. Erman. Relationships between amino acid sequence and backbone torsion angle preferences. *Proteins*, 55:992–998, 2004.

[16] 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. Jour.*, 85:2119–2146, 2003.

[17] R. Kuang, C. S. Leslie, and A. Yang. Protein backbone angle prediction with machine learning approaches. *Bioinformatics*, 20:1612–1621, 2004.

[18] M. Paluszewski and P. Winter. Protein decoy generation using branch and bound with efficient bounding. *Alg. Bioinf.*, pages 382–393, 2008.

[19] L. Perskie, T. O. Street, and G. D. Rose. Structures, basins and energies: A deconstruction of the protein coil library. *Prot. Sci.*, 17:1151–1161, 2008.

[20] N. Qian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202:865–884, 1988.

[21] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *J. Mol. Biol.*, 7:95–99, 1963.

[22] B. Rost. Twilight zone of protein sequence alignment. *Prot. Eng.*, 12:85–94, 1999.

[23] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, 232:584–599, 1993.

[24] Oliver Sander, Ingolf Sommer, and Thomas Lengauer. Local protein structure prediction using discriminative models. *BMC Bioinformatics*, 7(1):14, 2006.

[25] B. L. Sibanda, T. L. Blundell, and J. M. Thornton. 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.*, 206:759–77, 1989.

[26] Y. Zhang, S. Wu, and J. Skolnick. Ab initio modeling of small proteins by iterative tasser simulations. *BMC Biology*, 5:17+, 2007.

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

4.3 Paper: Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction

This paper is submitted to the 9th international conference on Computational Systems Bioinformatics, CSB 2010, to be held at Stanford University in August 2010. It is joint work with Rasmus Fonseca and Pawel Winter.

Ranking Beta Sheet Topologies with Applications to Protein Structure Prediction

Rasmus Fonseca[*], Glennie Helles[*] and Pawel Winter[*]

Abstract

One of the main reasons why ab initio protein structure prediction does not perform very well is the inability of structure predictors to reliable identify and model long-range interactions between amino acids. One way of handling this is to predict which β-strands form hydrogen pairs, ie. the β-topology. Since there is no reliable way of doing this, we generate all potential β-topologies for a set of proteins, and use two known β-topology scoring methods to rank them. An investigation of the rank-ordered list of potential β-topologies is performed, which has not previously been done. It is concluded that particularly one of the methods 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 assume that all potential β-topologies can be enumerated and scored for large proteins. We suggest an enumeration scheme that addresses this problem and show that native-consistent β-topologies are often among the top-ranked. Furthermore, prediction of β-topologies relies heavily on the correct identification of strands, so we present a method for dealing with the inaccuracies of secondary structure predictors. 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 constrained using top-ranked β-topologies as they are very likely to contain native or native-consistent β-topologies.

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

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

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 [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 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) between paired β-strands within 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 native β-topology could be correctly predicted, it would greatly improve the solutions to the PSP problem [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*, 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 here as the *topology scoring method*, is to enumerate all *potential 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 native [13], but the topology scoring method has also been used to filter decoy sets from Rosetta [19].

We enumerate all potential β-topologies and use the pair scoring method of Cheng and Baldi [16] and the topology scoring method of Ruczinski et al. [19] 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 potential 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 potential β-topologies as constraints (one at a time), a set of decoy structures can be constructed. At least one of these decoy structures will very likely be of high quality.

There are three serious problems with this approach. First of all, the correct secondary structure has been assumed known. The 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 underpredict β-strands. In such cases, the native β-topology will not 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 8 β-strands are predicted.

In order to deal with these problems, the notion of the *secondary structure assignment* (SSA) is introduced. This is a classification of each amino acid as either helix, strand or coil. For example, PSIPRED [1], one of the best secondary structure predictors, assigns to each amino acid the probability of it being either in a helix, in a strand and in a coil. Amino acids with their strand probabilities being higher than both helix- and coil probabilities are classified as belonging to β-strands. The same goes for helix and coil. The output of such a predictor is referred to as a *predicted SSA*.

To deal with the problem of over- and underprediction of strands in the predicted SSA, we use the probability levels for strands to generate a set of *strand candidates*. A subset of these strand candidates can be picked out and used to generate strands in a SSA. A list of *potential SSAs* is generated using every possible subset of strand candidates. The probability levels are used to calculate a score for every potential SSA and, as with β-topologies, they are scored and ranked.

The problem of combinatorial explosion is dealt with by introducing two limitations when generating the set of SSAs. First, only the 15 strand candidates with highest average strand-probability levels are used. Secondly, only potential SSAs with up to 7 strands are generated. Since only SSAs with 7 strands or fewer are generated, there are proteins with many strands whose native β-topology cannot be generated. However, enumerating potential SSAs and β-topologies is still relevant for such proteins. To illustrate this, the concepts of *native-respecting SSA* and *native-respecting β-topology* are defined. A native-respecting SSA is a SSA where every strand is present in the native SSA as well (though the native SSA may have more strands). Similarly, a native-respecting β-topology is a topology where every β-pair 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 SSA with up to 7 strands can always be found among the potential SSAs. 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, it is still a valid constraint that can reduce the search space significantly.

The results reported in this paper are highly relevant for 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 [21, 15].

2 Methods

In the following two subsections the methods for generating potential β-topologies and for calculating scores for these are described. Next it is described how potential SSAs are generated and how scores are assigned to each of them. The last two subsections describe how to compare both potential SSAs and β-topologies and which datasets are used to assess the methods.

Generating potential β-topologies

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

Given a SSA and the corresponding strands, all possible β-topologies can be generated. A β-topology generated from a SSA 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 $(i < j) \wedge (a_{ij} = 1)$. 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:

1. No strand, i, is paired to itself: $a_{ii} = 0$.
2. No pair of strands, (i, j) is paired both parallel and antiparallel: $a_{ij} + a_{ji} < 2$.
3. Every strand, i, is paired with at least one and at most two partners: $0 < \sum_{j=1}^{m} (a_{ij} + a_{ji}) < 3$.[1]

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.

For $m = 2, 3$ only β-topologies containing a single sheet is generated. For $m = 4, 5$ topologies with both one and two sheets are generated, and for $m = 6, 7$ topologies with up to 3 sheets are generated. Table 1 shows the number of valid β-topology-matrices for up to 7 strands. The number

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

Table 1: Number of matrices for $m \in [2, 7]$.

of β-topology-matrices grows exponentially with m and it is infeasible to enumerate all β-topology-matrices for $m \geq 8$. Later, it will be discussed how to cope with this combinatorial explosion.

[1] A few proteins have β-strands that form pairs with more than 2 partners which violates this rule.

The definition of a valid β-topology-matrix corresponds largely to the definition of 'overall' sheet configuration' used in [22] and 'β-sheet topology' from [16]. It is a representation of the β-topology that does not specify the individual amino acid pairs but focuses on the overall description of the β-pairs in the protein.

Assigning scores to β-topologies

Two methods for assigning scores to β-topologies have been examined: The topology scoring method [22, 19] and the pair scoring method [16].

The topology scoring method, as described in [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 [22] the probabilities of individual β-sheet topologies are combined with two more features, the number of sheets and number of crossings (consecutive β-strands in different β-sheets), to assign a probability to the entire β-topology.

The pair scoring method is based on the neural network from [16]. The network returns a pseudo-energy for each pair of amino acids in different strands. 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.

Generating potential SSAs

To ensure that the β-topology of a proteins native structure can be represented it is important that the placement of all strands are identified correctly. PSIPRED can be used to predict the placement of strands. It produces three probability levels for each amino acid, $a \in [1, n]$, describing the probability of a being either helix (pH_a), strand (pE_a) or coil (pL_a). If $pE_a > pH_a \wedge pE_a > pL_a$ then amino acid a is classified as strand. This method often fails to predict a strand entirely or predicts a strand where there isn't one. We have observed, however, that when it fails to predict a strand there is often a hilltop in the pE-levels of the amino acids. A set of *strand candidates*, representing possible placements of strands, are therefore generated around hilltops in the pE-plot (see Figure 1). Each candidate, $i \in [1, m_{\text{sc}}]$, is defined by the indices of its first and last amino acids: (s_i, e_i).

A potential SSA can be generated from a subset of strand candidates. When

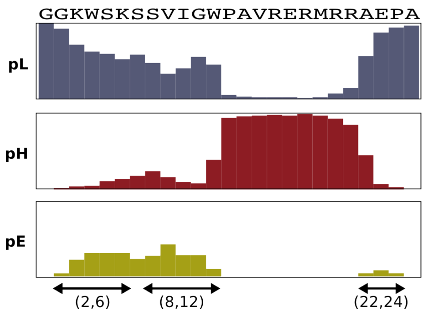
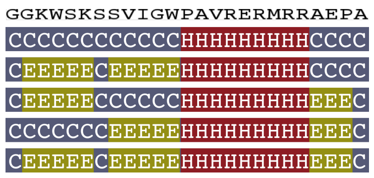


Figure 1: Probability levels for coil, helix and strand (pL, pH and pE) predicted for 1ZEC using PSIPRED [1]. The bottom row illustrates three strand candidates generated using the probability levels for strand and identified by their start and end-indices, (s_i, e_i).

a strand candidate, i, is part of this subset, all amino acids from s_i to e_i inclusive are assigned strand status. Since the results of both the topology scoring method and the pair scoring method depends on the assignment of helices as well, the status of the remaining amino acids is set to helix if $pH_a > pL_a$ and to coil otherwise.

Since a single SSA is generated using a subset of strand candidates, all the potential SSAs are generated by using all possible subsets of strand candidates. Potential SSAs with one strand, however, are omitted, as valid β-topologies must have at least 2 strands or none at all. Figure 2 illustrates all the possible SSAs that can be generated using the coil and helix probability levels and strand candidates from Figure 1.



```
GGKWSKSSVIGWPAVRERMRRAEPA
CCCCCCCCCCCCHHHHHHHHHCCCC
CEEEEECEEEEEHHHHHHHHHCCCC
CEEEEECCCCCCHHHHHHHHHEEEC
CCCCCCCEEEEEHHHHHHHHHEEEC
CEEEEECEEEEEHHHHHHHHHEEEC
```

Figure 2: All possible SSA's for 1ZEC using pL, pH and the strand candidates from Figure 1.

Three improvements are performed to ensure that better strand candidates are generated. The first is a smoothing of the pE-plot which is introduced by setting

$$pE_a \leftarrow c \cdot pE_{a-1} + (1 - 2c) \cdot pE_a + c \cdot pE_{a+1} \tag{1}$$

where $c = 0.25$. Also, pE-levels are reduced to 0 if they are below a threshold of 0.05. Different values of c and the threshold have been tested, but the selected ones were the best for ensuring that the native SSA was included in the generated potential SSAs. Both improvements are performed before generating strand candidates from 'hills'. The final improvement is performed after strand candidates are generated. If two strand candidates touch each other, i.e., $e_i = s_{i+1}$, then a small space is introduced by setting $e_i \leftarrow e_i - 1$.

If m_{sc} strand candidates are identified then $2^{m_{\text{sc}}}$ possible SSAs can be generated. The m_{sc} of these that have a single strand are not considered valid. As described previously, β-topologies are not generated for more than 7 strands, so a total of

$$1 + \sum_{s=2}^{\min(7,m_{\text{sc}})} \binom{m_{\text{sc}}}{s} \tag{2}$$

valid SSAs are generated. The first term is added because the SSA that has no strands is considered valid as well. For $m_{\text{sc}} > 15$ the number of valid SSAs is so large that it takes too long to execute the experiments. We therefore only consider $m_{\text{sc}} \leq 15$. If there are more strand candidates in a protein, we exclude those with the lowest average pE-levels. This can, at most, result in 15369 valid SSAs

Assigning scores to potential SSAs

The pE-levels are used to calculate a score for every potential SSA. First, each strand, $i \in [1, m]$, is identified by its start and end-indices (s_i, e_i) and the average pE_a value for that strand is calculated as

$$\langle pE \rangle_i = \frac{1}{e_i - s_i + 1} \sum_{a=s_i}^{e_i} pE_a \tag{3}$$

The score of a SSA is then

$$\frac{1}{m} \sum_{i=1}^{m} \langle pE \rangle_i \tag{4}$$

Comparing both SSAs and β-topologies

Before evaluating the generated SSAs and β-topologies, it is defined what it means that two SSAs match, one SSA respects another, two β-topologies match and a β-topology respects another.

Two strands, i and j, from different SSAs are said to overlap if any part of the interval $[s_i, e_i]$ overlaps $[s_j, e_j]$. Two SSAs *match* iff there exists a pairing of every strand in the first to every strand in the second such that each pair of strands overlap. An SSA is furthermore said to *respect* another SSA iff there exists a pairing of every strand in the first to a subset of strands in the second such that each pair of strands overlap. This definition will prove useful because potential SSAs that respect the native SSA can be considered 'almost native'. Figure 3 illustrates how SSAs are compared.

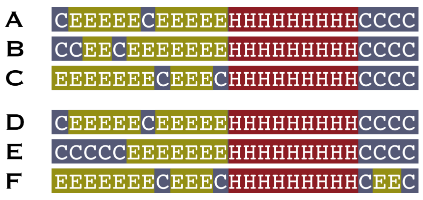

A
CEEEEECEEEEEHHHHHHHHHCCCC
B
CCEECEEEEEEEHHHHHHHHHCCCC
C
EEEEEEECEEECHHHHHHHHHCCCC
D
CEEEEECEEEEEHHHHHHHHHCCCC
E
CCCCCEEEEEEEHHHHHHHHHCCCC
F
EEEEEEECEEECHHHHHHHHHCEEC

Figure 3: Examples of comparing SSAs. A, B and C all match each other. D and E respects F but F neither respects nor matches any of the other.

A β-topology is given by a SSA 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 SSAs match and they have identical β-topology-matrices. Note that if the SSAs match then the β-topology-matrices will always be of the same dimension. One β-topology, with matrix $[a_{ij}]$, is said to *respect* another, with matrix $[a'_{ij}]$, iff:

1. Its SSA respects that of the second
2. $a_{ij} = 1 \Rightarrow a'_{kl} = 1$ where i and k are indices of strands that overlap, and j and l are indices of strands that overlap

Figure 4 illustrates how β-topologies are compared.

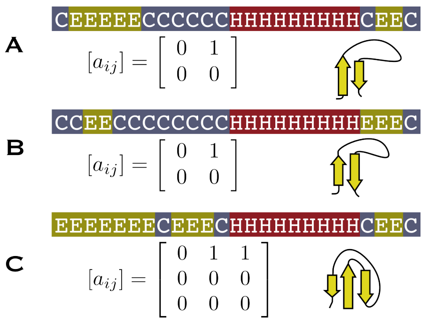

CEEEEECCCCCCHHHHHHHHHCEEC
A
$[a_{ij}] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$
CCEECCCCCCCCHHHHHHHHHEEEC
B
$[a_{ij}] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$
EEEEEEECEEECHHHHHHHHHCEEC
C
$[a_{ij}] = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Figure 4: Examples of comparing β-topologies. A and B match each other and they both respect C. C neither respects nor matches any other.

Datasets

For evaluating how well the scoring of SSAs and β-topologies is we generate two datasets. The first is made up of all the chains from PDBSelect25 2009 [23] that contains strands. This corresponds to 3305 out of 4423 chains (75%). Not all the numbers for recreating the topology scoring method was available in [22], so the dataset was split into a test-set, the *PDB test-set*, of 161 randomly chosen chains containing between 2 and 7 strands, and the remaining was used as a training-set.

We did not check if the PDB test-set contain proteins that have been used to train either PSIPRED or the neural networks in [16]. We speculate, however, that this does not affect the results significantly. To justify this 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 PDBSelect25, but it is certain that none of these proteins are part of the training-sets for PSIPRED or the neural networks in the pair scoring method. The use of a CASP8 test-set also gives an indication of the practical applicability of our method. At CASP8 there were 112 targets, but 14 contained no strands, so the CASP8 test-set consists of 108 protein chains (89%) that all have sheets. 53 of the these have between 2 and 7 strands and the majority of the rest contain between 8 and 12 strands.

The position of strands were identified according to the *SHEET*-records, columns 23-26 and 34-37, in the PDB-files. The *SHEET*-records, additionally, contain information about β-pairs, but they often have errors. We therefore decided if two strands were paired based on the pair-wise distances between amino acids in the two strands. If the largest of the three minimal pairwise distances was less than 6Å then the strands was said to be paired. A direction vector for each strand was used to determine parallelity. If the dot-product of two strands direction vectors was larger than 0 then they were defined to be parallel, otherwise antiparallel. The direction vector of a strand is the vector from the position of the first C_α atom to the last within the strand (going from the N-terminal to the C-terminal).

3 Results and discussion

The primary tool for analyzing the scoring methods for potential SSAs and β-topologies is a *rank-plot*. The rank-plot for potential SSAs of some protein shows the rank of each SSA (x-axis) and their score (y-axis). The SSAs are sorted by non-increasing score. The rank-plot is therefore a monotonically non-increasing curve representing scores of all potential SSAs (see Figure 5). The first potential SSA that match the native SSA (the *native-matching SSA*) is highlighted using a circle and SSAs that respect the native (*native-respecting SSAs*) are highlighted using crosses.

A rank-plot for potential β-topologies shows the rank and score for all the topologies, given a specific SSA (see Figure 6). Only a single β-topology can match the native and only topologies with 2 sheets or more (more than 3 strands) can respect (and not match) the native β-topology. These 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 SSAs and β-topologies will be the primary tool for reporting results.

Ranking β-topologies using native SSA

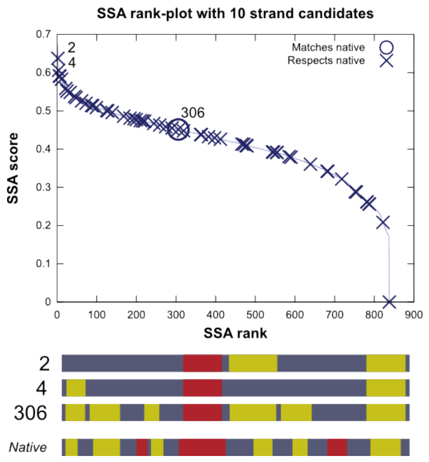


Figure 5: The SSA rank-plot for the six-stranded protein 1I8N. The native SSA has rank 306. However, potential SSAs with ranks as low as 2 and 4 respects the native, and will both be used to generate β-topologies that respects the native.

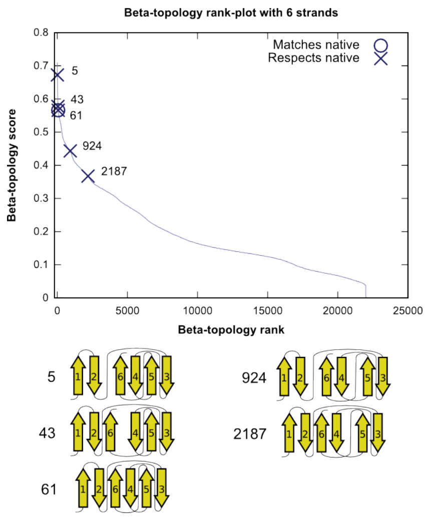


Figure 6: The β-topology rank-plot for the six-stranded protein 1I8N. The native SSA has been used, and the scores are calculated using the pair scoring method. The native β-topology has rank 61 (native rank), but the β-topology with rank 5 respects the native (respecting rank), and thus provides a constraint that is nearly as good as the native. All topologies that either match or respect the native are highlighted and shown below the plot.

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

An important question when considering the applicability of enumerating

β-topologies is: How many of the top-ranked β-topologies does one have to enumerate before the native-matching is found? Figure 7 shows how many proteins (percentage) that have the native-matching β-topology among the top-ranked. The figure illustrates this for both scoring methods – the topol-

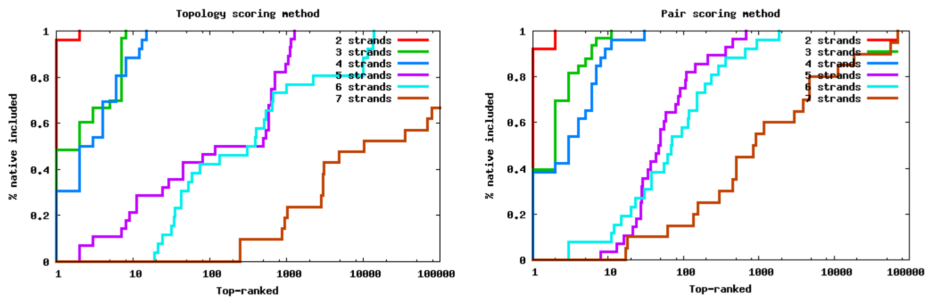


Figure 7: Percentage of native-matching β-topologies among the top-ranked potential topologies using the pair scoring method and the topology scoring method. The x-axis shows the number of top-ranked topologies on a logarithmic scale.

ogy scoring method (top) and the pair scoring method (bottom). Individual curves are generated for proteins containing the same amount of strands. If it is assumed that the native SSA is known (or predicted correctly) then 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 going through just a relatively small number (hundreds) of β-topologies gives a constraint for the PSP problem that corresponds completely to the native-matching. We implemented a third method for scoring β-topologies that was based on hydrophobic packing [26]. This method, however did not perform better than either the topology scoring method or the pair scoring method, so the results were not included. The topology scoring method performs good, 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 2 shows statistics for the rank of the native-matching β-topology.

Strands	2	3	4	5	6	7
Proteins	26	33	26	28	27	20
Avg. rank	1.08	2.55	4.77	104	213	8850
Median rank	1	2	3	49	69	905

Table 2: Average and median ranks of native-matching β-topologies in PDB test-set.

There were 122 targets at CASP8 and 53 of these had between 2 and 7 strands. This means that, assuming the native SSA can be correctly predicted, the native β-topology can be generated for 43% of the protein at CASP8. Only 10 proteins have between 2 and 4 strands, so statistics generated for $m = 2, 3$ and 4 are not significant. Table 3 shows statistics for the rank of the native-matching β-topology. Comparing these numbers to those

Strands	5	6	7
Proteins	13	11	16
Avg. rank	67.5	872	4240
Median rank	16	149	768

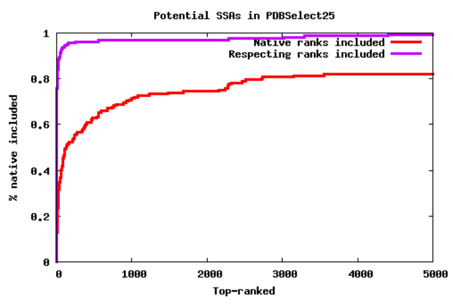
Table 3: Average and median ranks of native-matching β-topologies in CASP8 test-set.

for the PDB test-set in Table 2, it is observed that the average and median native rank is higher for the proteins with 6 strands, but notably lower for those with 5 and 7 strands. This implies that even if some of the proteins in the PDB test-set are in the training-set for the neural networks in [16], it does not affect these results. 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.

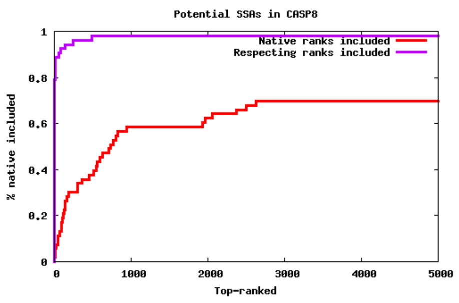
Ranking potential SSAs

PSIPRED was downloaded from `http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/`, and used to generate pH, pE and pL levels for all proteins in the PDB test-set. From the pE-levels, strand candidates are identified and potential SSAs generated. For every potential SSA, a score is calculated using the pE-levels, and a rank-plot is generated for every protein. The number of potential SSAs that one has to enumerate before the native-matching SSA is encountered is shown in Figure 8. The red curve converges on $\approx$ 81% after 3000 potential SSAs (out of the $\approx$15000), which indicates that for only 19% of the proteins in the PDB test-set, a potential SSA that matches the native is not 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 less than 1000 potential SSAs. Since there are generated a total of up to 15369 potential SSAs it is observed that, for a majority of the proteins the native SSA can be found among the top 7% of the generated SSAs. Using only the top-200 ranked potential SSAs a native-respecting SSA can be found for more than 95% of the proteins. This is shown in the purple curve in Figure 8a.

The same experiment is repeated for the proteins in the CASP8 test-set that have between 2 and 7 strands. The results are shown in the bottom of Figure 8b. These curves follow roughly the same trend as for the PDB test-set. This indicates that, even if some proteins in the PDB test-set are in the training-set of PSIPRED, it does not affect our results.



(a)



(b)

Figure 8: Percentage of proteins for which a the native-matching SSA is included in the top-ranked SSAs (red curve). The purple curve shows the percentage of proteins for which the native-respecting SSA is included among the top-ranked. In (a), only the proteins from the PDB test-set with 2 to 7 strands are included. In (b), only the proteins from the CASP8 test-set with 2 to 7 strands are included.

PSIPRED typically uses BLAST [27] to locate sequence profiles that can help improve the secondary structure prediction. There is an option that disables this feature, but it decreases the accuracy of PSIPRED. We found, however, that when disabling BLAST the pE-levels more often resulted in good strand candidates that ensured the native SSA was among the potential SSAs. We therefore choose to disable BLAST. This observation indicates that creating a secondary structure predictor specifically for identifying strand candidates could further improve our result and increase the ratio of proteins for which the native-matching SSA could be identified.

Combining potential SSAs and potential β-topologies

This subsection seeks to determine the applicability of enumerating both potential SSAs and potential β-strands. Since the CASP8 test-set contains proteins that state-of-the-art PSP methods are benchmarked on we will use this test-set for the following experiments. The combinatorial explosion of β-topologies is dealt with by primarily analyzing the native-respecting β-topology. This ensures that the experiment can be run on proteins with more than 7 strands.

m	2	3	4	5	6	7
$B(m)$	2	11	30	700	1900	70000

Table 4: $B(m)$ for different values of m. Can also be read from Figure 7.

Min. m	μ(SSA)	$\mu_{\frac{1}{2}}$(SSA)	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

Table 5: Combining potential SSAs and β-topologies for the 108 CASP8 proteins with at least 2 strands. μ(SSA) and $\mu_{\frac{1}{2}}$(SSA) denotes the average and median rank of the first native-respecting SSA 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.

The experiment seeks to determine how many β-topologies it is necessary to enumerate to find the native-respecting β-topology without assuming that the native SSA is known in advance. It can be seen in Figure 7 that, to be sure that a native-matching β-topology is among the top-ranked for an SSA with m strands, it is necessary to include the $B(m)$ top-ranked β-topologies where $B(m)$ is defined in table 4. For each of the 108 proteins in the CASP8 test-set, the following experiment is performed: The set of potential SSAs is generated, scored and ranked. Starting from the top-ranked potential SSA, 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 SSAs until the first native-respecting β-topology is encountered. The number of examined β-topologies is then reported. The average and median of this number is shown in the second row of Table 5. There is a huge difference between the average number of β-topologies that has to be examined ($\approx$ 80000) and the median (44). This indicates that only a limited number of outliers needs to have many 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 2 strands. This does not provide a very strong constraint on the PSP problem. We therefore repeat the experiment above, but for potential SSAs with at least 3, 4 and 5 strands. As a result $\approx$ 10000, $\approx$ 22000 and $\approx$ 240000 β-topologies respectively have to be enumerated for a majority of the proteins before a native-respecting topology is found. Although these numbers are high, it is still realistic to generate that many decoys in a PSP method. The total number of β-topologies is approximated by multiplying the total number of potential SSAs with the average number of examined β-topologies for each SSA. For most proteins 15 strand candidates are identified, which gives 15369 potential SSAs.

Multiplied by the average of $B(m)$ this indicates that the total number of generated β-topologies roughly $1.8 \cdot 10^8$.

The focus of this section has, so far, been solely on native-respecting β-topologies because these can be found for proteins containing any number of strands. This was ensured that useful results could be generated even for proteins with more than 7 strands. The above experiment was repeated for proteins with 7 strands or fewer and the number of examined topologies were reported only when the native-matching β-topology was examined. The average and median number of topologies that had to be examined were around $13,900,000$ and $4,800,000$ and this could only be done for 33 of the 53 proteins (62%). While these numbers are rather large, a PSP method that efficiently takes advantage of β-topologies, such as [14], will be able to go through this many topologies in a limited amount of time. Furthermore, for a few proteins (3DFD, 3DED, 3DEX, 2KDM and 3DO8), the native β-topology was found after only examining a few thousand topologies.

4 Conclusions and future work

We have presented a method to enumerate and rank potential β-topologies for proteins with up to 7 strands using two different scoring methods; the pair scoring method and the topology scoring method. The pair scoring method was shown to outperform the topology scoring method.

If we do not know the correct secondary structure assignment (SSA) in advance, we first use output from PSIPRED to generate potential SSAs with up to 7 strands and then rank these assignments. The results show that the native SSA is among the top 7% highest ranked SSAs for the majority of proteins.

SSAs are then used to generate potential β-topologies. Given the correct SSA we found 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 SSAs, we found that non-trivial (i.e. more than two strands) native-respecting β-topologies can be found within the top 10000 highest ranked 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 [19] and the pair scoring method [16]. Features and concepts from other sources such as [28, 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.

The results indicate that a relatively large number of SSAs has to be examined before the native SSA is located. The method used for scoring potential SSAs is very simple. We therefore believe that a huge improvement of our results could be achieved by refining the scoring of potential SSAs using, for instance, machine learning methods like neural networks or support vector machines. As mentioned previously, a secondary structure predictor that overpredicts strands could also help to ensure that the native SSA was among the potential SSAs for more than 81% of the proteins.

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.

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:i75–84, 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] Nicholas Hamilton and Thomas Huber. An introduction to protein contact prediction. pages 87–104. 2008.

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

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

[24] ShuoYong Shi, Jimin Pei, Ruslan I. Sadreyev, Lisa N. Kinch, Indraneel Majumdar, Jing Tong, Hua Cheng, Bong-Hyun Kim, and Nick V. Grishin. Analysis of casp8 targets, predictions and assessment methods. *Database*, 2009(0):bap003+, April 2009.

[25] Michael L. Tress, Iakes Ezkurdia, and Jane S. Richardson. Target domain definition and classification in casp8. *Proteins*, 77 Suppl 9(S9):10–17, 2009.

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

[27] SF Altschul, TL Madden, AA Schaffer, J Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.*, 25(17):3389–3402, 1997.

[28] Jennifer A Siepen, Sheena E Radford, and David R Westhead. Beta edge strands in protein structure prediction and aggregation. *Protein Sci*, 12(10):2348–59, Oct 2003.

Chapter 5

Other work

5.1 Background

Genetic algorithms aims at mimicking evolution in order to find (near-)optimal solutions, but how well do they actually do this? Interestingly, genetic algorithms are primarily used in optimization and have in fact never really been used as a simulation tool for population geneticists. In lack of any literature supporting this notion, I set out to determine if this was simply a coincidence or because genetic algorithms are simply too coarse grained to be of practical use in population genetics.

5.2 Paper: Simulating Evolution of *Drosophila melanogaster Ebony* Mutants Using a Genetic Algorithm

This paper was presented at evoBIO 2009 and published in "LNCS: Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics" [7].

Simulating Evolution of *Drosophila melanogaster Ebony* Mutants Using a Genetic Algorithm

Glennie Helles

Abstract

Genetic algorithms are generally quite easy to understand and work with, and they are a popular choice in many cases. One area in which genetic algorithms are widely and successfully used is artificial life where they are used to simulate evolution of artificial creatures. However, despite their suggestive name, simplicity and popularity in artificial life, they do not seem to have gained a footing within the field of population genetics to simulate evolution of real organisms – possibly because genetic algorithms are based on a rather crude simplification of the evolutionary mechanisms known today. However, in this paper we report how a standard genetic algorithm is used to successfully simulate evolution of *ebony* mutants in a population of *Drosophila melanogaster* (D.*melanogaster*). The results show a remarkable resemblance to the evolution observed in real biological experiments with *ebony* mutants, indicating that despite the simplifications, even a simple standard genetic algorithm does indeed capture the governing principles in evolution, and could be used beneficially in population genetics studies.

1 Introduction

Darwin's book "On the Origin of the Species" was published in 1859 and even though evolution was already generally a recognized phenomenon amongst scientists at the time, Darwin offered a new way to explain evolution that has since been broadly accepted by the scientific community.

Genetic algorithms can generally be thought of as a formalization of Darwin's evolutionary theory. Originally formulated by John Holland in the 1960's, genetic algorithms are a set of heuristics that are perhaps mostly known for their ability to find optimal or near-optimal solutions in large (possibly infinitely large) search spaces [4, 7] and their use in artificial life

[5]. They come in many different shapes and forms, but the underlying framework involving an encoding-, selection- and breeding-strategy is the same for all of them.

In this experiment we use a genetic algorithm to simulate a population of *Drosophila melanogaster* (D.*melanogaster*), also known as the common fruit fly. The fly is with its approximately 13.600 genes a relatively complex animal with a fairly short life cycle of roughly 30-40 days [3], although flies kept in laboratories usually have a slightly shorter life span[3]. This makes the fruit fly ideal for studies of population genetics and it is indeed widely used for just that [10].

One of the genes that has been throughly studied is the *ebony* gene [10], which aside from being associated with the color of the fly, influences both vision and movement and has a direct influence on the breeding success of the fly. Furthermore, only approximately 80% of *ebony* mutants ever hatch giving the mutants a natural disadvantage [10].

Each fly has two instances of each chromosome, which gives rise to three distinct *ebony* genotypes: the mutant genotype (e/e) has a mutation in the *ebony* gene on both chromosomes, the heterozygous genotype (+/e) has a mutation in the *ebony* gene on only one of the chromosomes while the wild type genotype (+/+) has no mutation in the *ebony* gene. The *ebony* gene is a recessive gene, which generally means that only flies with the mutant genotype will show the mutant characteristics. In practice, however, many of the heterozygous flies can easily be distinguished upon visual inspection and mating studies also shows that the flies are definitely able to distinguish between all three genotypes[8]. From a population geneticist's point of view, the *ebony* gene is interesting because hatching along with the ability to successfully move around and breed are generally considered extremely important for a fly, if the genes are to be preserved in future generations. Aside from their lower hatching success, the ability of the *ebony* mutant to move around is also impaired compared to both the wild type fly and especially the heterozygous fly, just as the mutants eyesight is quite limited [10]. A male fly's ability to move around quickly is thought to be an attractive characteristic among female flies, and good vision will naturally allow the flies to spot the most attractive mating partner from a longer distance.

However, the male mutant fly has a larger mating capacity (it can inseminate more females in a 24 hour period) than the wild type counter part [8, 9] and the female mutant flies tend to be less selective when being courted by a male fly – possibly because of their impaired vision. Heterozygous male flies have the largest mating capacity of all the genotypes, which would seem to give this genotype an advantage, but the heterozygous female flies on the other hand accepts the fewest mating invitations, thereby apparently canceling out the advantage. Biological experiments, wherein the initial population contains 25%, 50% and 75% mutants respectively, have shown that the frequency of mutants always drop rapidly during the first 5-10 generations, but then stabilizes (see [8, 9] and Figure 1 A).

A number of systems for simulating and predicting allele frequencies of genes exists. Very simple systems based on Wright-Fisher populations assume neutral development and are thus too imprecise for studying real systems. Other non-neutral systems that use Markov chains have been applied [2] but to our knowledge genetic algorithms have for some reason, despite their otherwise suggestive name, not previously been used in population genetics simulations. The appeal of using a genetic algorithm for a population geneticist is that the terminology is highly familiar and thus very intuitive. We realize that genetic algorithms have been developed and extended much over the years as our understanding of evolution has increased [1], but this type of experiment relates to the very fundamentals of evolution and we thus believe that, aside from being very simple, the original genetic algorithm is quite adequate.

2 Methods

The genetic algorithm used for the simulation is constructed such that each individual (fly), referred to as a D.*simulation*, has two chromosome pairs explicitly modeled. The chromosome 3 pair, which includes the *ebony* gene, and the sex chromosome pair (X/Y), which determines the sex of the fly. All flies possess the following basic characteristics: they can fly, walk, see and breed. Furthermore, we have introduced a “resting period” for the male D.*simulation*, which is triggered by the mating procedure, to ensure that the simulated fly has the same mating capacity as the real fly. Female flies do not have a resting period, but they do have the ability to lay eggs. How far they can fly, walk or see depends on the genotype of the fly (see Table 1) just as it is the case for the real fly and likewise for the required resting period of the male flies and the selectivity of the female flies. During the resting period the male fly is unable to mate, but it may still move around. How many cells the simulated flies can move and see is based on our interpretation of how the different genotype compared to each other.

D.*melanogaster* goes through several stages before it becomes a real fly [3]. However, the egg stage, larval stage and pupa stage are for the most part uninteresting when looking at population genetics and D.*simulation* thus only has one stage (the egg stage) before it becomes a fly. Development from egg to fly for D.*melanogaster* take roughly 14 days and the length of the egg stage for D.*simulation* is thus chosen randomly between 12 and 16 days. The egg never moves. Like the D.*melanogaster ebony* mutant, a D.*simulation ebony* mutant egg has only a 80% chance of ever hatching. Eggs that do not hatch are removed from the world. After hatching the D.*simulation* flies are sexually active after 12 hours and the female flies start laying eggs after 2 days. These values are equal to those observed in real experiments [3]. Real female flies may lay several hundreds of eggs – whether they are fertilized or not – over a period of approximately 10 days, but in order to control population size a female D.*simulation* lays only between 1 and 6 eggs. All unfertilized eggs are removed immediately from the environment. The complete life cycle of D.*simulation* including the egg stage is chosen randomly between 28 and 32 days. The initial population contains 100 flies and 20 eggs (50% of each sex). If the population grows to include more than 250 flies, a 1% chance of "sudden death" is introduced.

The simulation uses a 3-dimensional world containing 50 x 50 x 50 cells. Each cell can only hold one fly or egg. Each time step is equivalent to one hour and the flies can perform one movement per time step. How far a fly moves in each time step is chosen randomly, although it never exceeds the distance stated in Table 1. If a cell is already occupied by another fly a new movement and distance is chosen. Mating occurs at the end of each time step. Female flies are selected with a frequency that match their "courtship acceptance" and for each selected female fly a male fly is chosen by using a fitness-proportionate selection strategy known as a "roulette wheel" on the non-resting male flies that occupy cells within eyesight distance of the female. The Roulette Wheel ensures that male flies with higher fitness values are chosen more frequently than male flies with lower fitness values.

The breeding strategy uses only mutation (not crossover), as the simulation is only focused on the *ebony* gene. Spontaneous mutation of the *ebony* gene occurs with a frequency of 8x10^{-4}. The mutation frequency is set 40 times higher than what is typically observed in real life to compensate for the inclusion of only one type of *ebony* mutants, as opposed to the 40 different mutations to the *ebony* gene that has been identified in the real fly.

	+/+	+/e	e/e
Walk (cells)	5	5	4
Fly (cells)	12	12	8
Sight (cells)	5	5	3
Mating capacity (24 hours)	3	6	4
Courtship acceptance (%)	60	50	65

Table 1: For each genotype (+/+ = wild type, +/e = heterozygous and e/e = mutant) is indicated the maximum flying and walking distance, how far they can see, the mating capacity of the males and the courtship acceptance frequency of the females.

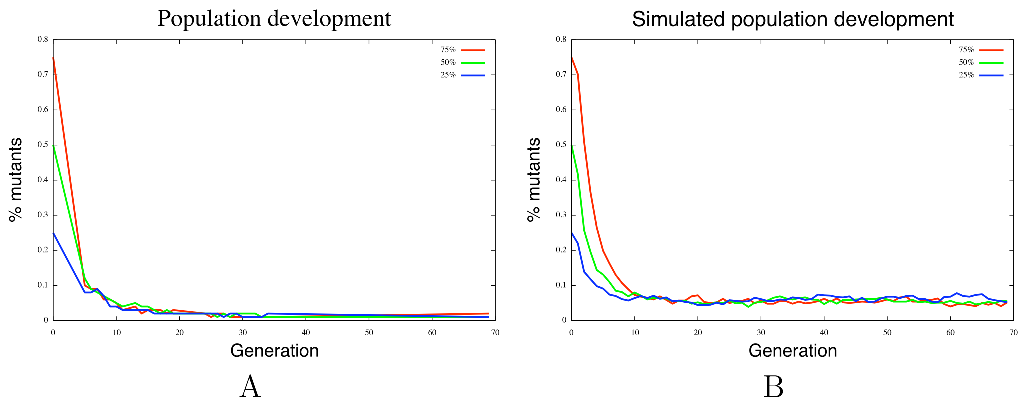


Figure 1: The results taken from the real experiments compared to the results from the simulation. The simulated result shown for each start population of 25%, 50% and 75% mutants is the average of the 10 runs.

3 Results

The fitness values upon which these results are based are listed in Table 2. They are the result of simply adding the values from Table 1, except "courtship acceptance", which is purely a female trait that does not influence the male fitness value. The simulation is run 10 times of 50000 time steps (≈ 5.7 years) for each start population of 25%, 50% and 75% mutants respectively. Due to the stochastic aspect of genetic algorithms, different results may be obtained for each run and running the algorithm multiple times was thus done.

Both the results from real experiments with D.*melanogaster* and the results from the simulation are presented in Figure 1. The results from the real experiment are taken directly from [10]. The figure shows that the frequency of mutants in both the D.*melanogaster* and D.*simulation* population drop very rapidly initially and then stabilizes after roughly 10-15 generations. As can be seen, development of the gene in the simulated population matches the real experiment close to perfectly.

It should of course be emphasized that selection of flies are based on the very simple rules that are defined in section 2 and they remain the same throughout the simulation. The composition of genotypes in the population

	Fitness
+/+	25
+/e	28
e/e	19

Table 2: Each of the fly's characteristics contributes to the overall fitness value, but varies for every genotype (+/+ = wild type, +/e = heterozygous and e/e = mutant).

at a given time is not considered and does not in any way affect which flies are selected for breeding. In other words, we do not have any special rules that are applied if mutants appear to either take over the population or die out.

4 Discussion

The main focus of this experiment is to determine if a standard genetic algorithm is able to successfully capture the fundamental aspects of evolution as can be observed in population genetics experiments. By inspecting the mutant frequency from experimental results with D.*melanogaster* (Figure 1 A), two things stand out: the initial drastic drop in mutant frequency and the following stabilization. Both traits are clearly observed in the simulated population (Figure 1 B). One could fear that the mutants would either completely take over the population or alternatively simply die out, as indeed seems very likely when observing the initial drastic drop, but the pattern resembles the pattern observed in nature. With the settings given in Table 1 and 2 the genetic algorithm does thus appear to very successfully capture the governing aspect of evolution and would in fact seem to be an obvious choice for simulating population genetics.

Another interesting observation is that, in Søndergaard [9] they describe how females tend to pick differently depending on the male compositions in the population, but we do not actually have to explicitly consider that in the simulation. The females follow that same rules at all times during the simulation regardless of the male composition in the population, and the behavior thus seem to emerge naturally simply from each individual following a few simple rules. This phenomenon is also known in swarm intelligence where simulation have clearly demonstrated that the seemingly intelligent movement of flocking animals in fact emerges from the simple set of rules that each animal follow [6].

When genetic algorithms are as accurate, as seen here, they can be used not only to simulate evolution but also to investigate underlying biological aspects. It is for example quite interesting to note that although the *ebony* gene is recessive, and one would thus expect that the heterozygous fly and the wild type fly should be viewed as equal, it proved to be imperative for the accuracy of the simulation to distinguish between the two genotypes. If such a distinction was not made the drop in mutant frequency was generally more linear, and most often the mutants died out before 50000 time steps.

We wish to emphasize that we do not suggest that simulation with a standard genetic algorithm can be used to simulate evolution of, say, the effect of a new mutation - the genetic algorithm relies heavily on prior knowledge that is relevant for calculating fitness of the fly. However, the effect of changing for instance the start compositions of flies (such as more mutants or fewer males) or even introducing impairments (such as restrictions on walking distance) could easily be studied using a simulation as this. Also, an analysis of exactly how much the fitness values can be varied while maintaining the characteristic development pattern could be done to help us understand how robust nature really is and indicate how big an advantage or disadvantage each genotype can have without altering the population development pattern. Running the computer simulation for 50.000 time steps (corresponding to $\approx$5.7 years or 70 generations) with our parameter settings takes just under 5 minutes on a 2.2GHz Intel processor.

As a final note, it should be mentioned, that this simulation is rather simple, as it involves only mutation as a biological operator. The other major biological operator, crossover, is not really used in this experiment, and we are thus not able to draw any conclusions about how well genetic algorithms would for instance simulate evolution of couple genes, but given the good results we have achieved here with relatively little effort, we are quite optimistic, and it would be interesting to attempt a more complicated simulation.

5 Conclusion

The results from experiments with D.*melanogaster* and the results from the simulated fly, D.*simulation*, bear a remarkable resemblance. Despite being based on a simplification of evolution as we know it, genetic algorithms do appear to be quite able to capture the fundamental aspects of evolution. The prospect of using genetic algorithms in population genetics where some knowledge about fitness have already been established thus look very good and further investigation into more complicated simulations of, for instance, coupled genes in the D.*melanogaster*, would be interesting to carry out.

References

[1] W. Banzhaf, G. Beslon, S. Christensen, J. A. Foster, F. Kps, V. Lefort, J. F. Miller, M. Radman, and J. J. Ramsden. From artificial evolution to computational evolution: a research agenda. *Nat. Rev., Gen*, 7:729–735, 2006.

[2] P. Fearnhead. Perfect Simulation From Nonneutral Population Genetic Models: Variable Population Size and Population Subdivision. *Genetics*, 174:1397–1406, 2006.

[3] A. J. F. Griffiths, J. H. Miller, D. T. Suzuki, R. C. Lewontin, and W. M. Gelbart. *An Introduction to Genetic Analysis.* Freeman, 1996.

[4] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1999.

[5] M. Mitchell and S. Forrest. Genetic algorithms and artificial life. *Artificial Life*, 1:267–289, 1994.

[6] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Comp. Graph.*, 21:25–34, 1987.

[7] S. Russel and P. Norvig. *Artificial Intelligence - a modern approach.* Prentice Hall, 1995.

[8] L. Søndergaard. Mating competition on artificial populations of Drosophila melanogaster polymorphic for ebony —. *Heraditas*, 103:47–55, 1985.

[9] L. Søndergaard. Mating competition on artificial populations of Drosophila melanogaster polymorphic for ebony II. *Heraditas*, 103:47–55, 1985.

[10] L. Søndergaard and K. Sick. Long-term frequencies of *ebony* in artificial Drosophila melanogaster populations kept in light and darkness. *Heraditas*, 103:57–61, 1985.

Chapter 6

Conclusions

The main focus of this thesis has been to devise and experiment with methods and algorithms that can improve *ab initio* protein structure prediction. Four of the five papers thus pertain to this problem. The fifth paper included in this thesis touches on a different problem within of the bioinformatics area, namely simulation of population genetics. Four of the papers included have been presented at international conferences and/or published in international journals. The fifth is currently in review.

In the thesis we have:

- presented an in-depth survey and comparison of existing methods for *ab initio* PSP. Due to the lack of standard protein test sets, performance comparison of existing algorithms found in the literature is far from simple, yet important if we want to find ways of improving the prediction methods. In the survey, we thus identify all parameters that may influence performance and compare the setting of these across recently published *ab initio* PSP methods.
- investigated parallel meta-heuristics for protein structure prediction and propose an iterative variant of the niche genetic algorithm (inGA). We show through experiments how the combination of parallelism, iterations and a highly elitist GA is clearly beneficial as it outperforms both the traditional niche GA and the parallel version of simulated annealing, known as parallel tempering, by finding lower energy structures much faster.
- showed how a neural network can be used to predict a probability distribution for coil residues that could be used to guide search algorithms to the most probable dihedral angle area for coil residues.
- devised a way to enumerate and rank potential β-topologies such that we are likely to encounter native or native-respecting topologies among the top ranked topologies.
- illustrated how a genetic algorithm can simulate evolution of *ebony* mutants in a population of Drosophila *melanogasters* (fruit flies) with very high accuracy.

We believe that all of our findings are of high value to the research community. For PSP, we have several ideas for extensions. Particularly our last work on predicting β-topologies has brought on a number of new ideas. Generating constraints is highly attractive, because it limits the solution space significantly, but it also poses a huge challenge for meta-heuristics. Currently, meta-heuristics simply cannot utilize the constraints properly to make the search faster. This is because the way meta-heuristics generate new solutions (by mutations or recombinations) will frequently take us to "illegal" areas of the solution space, i.e., areas where the constraints are not upheld. We thus need some mechanism to ensure that the meta-heuristics only visit the solution space where the constraints are indeed upheld. One of my main focus points for further research is contriving a method – possible a variant of a meta-heuristic – that can somehow make proper use of the constraints. I believe this would improve both the execution time and the solution quality of predicted structures.

Bibliography

[1] C. B. Anfinsen. Principles that governs the folding of protein chains. *Science*, 181:223–230, 1973.

[2] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. *Proc. Sec. Int. Conf. Comp. Mol. Biol.*, pages 30–39, 1998.

[3] Y. Choi and C.M. Deane. Fread revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, 78:1431–1440, 2009.

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

[5] F. Gratzon. *The Lazy Way to Success*. Soma Press, 2003.

[6] G. Helles. A comparative study of the reported performance of *Ab Initio* protein structure prediction algorithms. *J. R. Soc. Interface*, 5:387396, 2008.

[7] G. Helles. Simulating evolution of *Drosophila melanogaster Ebony* mutants using a genetic algorithm. *LNCS*, 5483:37–43, 2009.

[8] D. Katagiri, H. Fuji, S. Neya, and T. Hoshino. Ab initio protein structure prediction with force field parameters derived from water-phase quantum chemical calculation. *J Comp. Chem.*, 29:1930–1944, 2008.

[9] E. Kim, S. Jang, and Y. Pak. All-atom ab initio native structure prediction of a mixed fold (1fme): A comparison of structural and folding characteristics of various $\beta\beta\alpha$ miniproteins. *J. Chem. Phys*, 131:195102, 2009.

[10] C. Levinthal. Are there pathways for protein folding. *Ext. Jour. Chim. Phys.*, 65:44–45, 1968.

[11] G. Nicosia and G. Stracquadanio. Generalized pattern search algorithm for peptide structure prediction. *Biophys. Jour.*, 95:4988–4999, 2008.

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

Appendices

Appendix A

Improving search for low energy protein structures with an iterative niche genetic algorithm

Improving search for low energy protein structures with an iterative niche genetic algorithm

Glennie Helles

Abstract

In attempts to predict the tertiary structure of proteins we use almost exclusively metaheuristics. However, despite known differences in performance of metaheuristics for different problems, the effect of the choice of metaheuristic has received precious little attention in this field. Particularly parallel implementations have been demonstrated to generally outperform their sequential counterparts, but they are nevertheless used to a much lesser extent for protein structure prediction. In this work we focus strictly on parallel algorithms for protein structure prediction and propose a parallel algorithm, which adds an iterative layer to the traditional niche genetic algorithm. We implement both the traditional niche genetic algorithm and the parallel tempering algorithm in a fashion that allows us to compare the algorithms and look at how they differ in performance. The results show that the iterative niche algorithm converges much faster at lower energy structures than both the traditional niche genetic algorithm and the parallel tempering algorithm.

1 INTRODUCTION

Metaheuristics are known to perform well on high-complexity problems where the search space becomes too big for exhaustive search to be feasible. Prediction of the three-dimensional structure of proteins from their primary sequence alone, known as *ab initio* or textitde novo folding[1], is such a problem and metaheuristics are almost exclusively used to solve this problem [6, 9].

Proteins are made up by amino acids that are strung together like pearls on a string such that each amino acid is connected to its neighboring amino acid(s) via

[1]The term *ab initio* traditionally refers to prediction methods that start without any knowledge of any globally similar folds thereby setting them aside from homology modeling techniques. However, many so called *ab initio* methods do in fact use secondary structure prediction algorithms that are trained from knowledge of already known structures, or they use fragment assembly compiled from known structures. Some choose to refer to this as *de novo* prediction rather than *ab initio* prediction. The term *ab initio* will be used in this publication

a peptide bond, ω. However, despite the rigid nature of the peptide bond, atoms can in theory rotate almost freely around the two other backbone bonds – the N–C_α bond, ϕ, and the C_α–C' bond, ψ – which means that just like a pearl necklace, a protein can be folded up in infinitely many ways, which is the reason that protein structure prediction poses such a big problem. Fortunately, steric clashes between atoms in neighboring amino acids do impose a significant restraint on the flexibility of the ϕ and ψ angles actually observed for amino acids [10], but searching exhaustively for the structure with the lowest energy remains elusive.

Judging from the literature, the Monte Carlo variant known as Simulated Annealing appear to be the preferred meta-heuristic for *ab initio* structure prediction followed by Genetic Algorithms [6]. Both metaheuristics can be parallelized, and the parallel versions are generally believed to perform better in rugged energy landscapes like those associated with protein structure prediction [3]. Oddly enough, the parallel versions are nevertheless used to a much lesser extent than their sequential counterparts in the field of protein structure prediction. We speculate that that is mostly because the effect of the choice of metaheuristics has been paid little attention in this field that is notoriously haunted by many other fundamental problems. The most significant obstacle probably being finding an appropriate energy function that can be used to score a protein, which has by far received the most attention over the years.

To our knowledge there exists only one parallel version of the Simulated annealing algorithm known as the Parallel Tempering or Monte Carlo Replica Exchange algorithm [12, 3]. In Parallel tempering (PT) many simulations, or replicas, are started and run in parallel. The solutions are sampled in the same fashion as in the regular simulated annealing approach by making small alterations to the solution and accepting the change with a certain probability. However, instead of lowering the temperature like in the simulated annealing approach the simulations are run at different but steady temperatures throughout the search. Two replicas may be swapped with a probability that depends on both differences in energy and temperatures, such that a replica running at a lower temperature can be exchanged with a replica running at a higher temperature, thereby giving replicas a greater chance of overcoming local minima barriers.

For genetic algorithms there exists several parallel variants that generally offer significant improvements by converging faster at often better solutions than the non-parallel version. There are two major approaches to parallelizing genetic algorithms. One is often referred to as the master-slave model, where a single process (the master) controls the genetic algorithm, but uses a number of other processes (the slaves) to evaluate and possible breed the individuals. The slave processes are run in parallel.

The other parallelization paradigm frequently used is the niche model (also known as the island hopping or deme model). A niche genetic algorithm (nGA) is an implementation where several instances of a genetic algorithm are run in parallel, evolving sub-populations independently from each other (the different niches). At certain points during evolution individuals migrate to other niches and become part of the population of that niche. The major advantage of nGAs is that they not only allow evolvement of multiple solutions at the same time, they exploit the fact that different runs of the same genetic algorithm is likely to produce different suboptimal solutions, that combined are likely to yield better results. Like PT the advantage of nGA is expected to be more profound when the fitness landscape is very rugged.

PT and nGA, with N replicas and niches respectively, essentially requires N times more computational time than a single run of their sequential counterparts. However, with multi-core computers and CPU clusters being readily available to most researchers, they can be executed in parallel and the extra computational time required does thus not impose a problem. On top of that, PT and nGA generally search more efficiently and usually arrive at much better results, which makes the parallelization of these meta-heuristics an attractive feature indeed.

In this paper we propose an iterative variant of a nGA, called inGA (iterative niche genetic algorithm) for protein structure prediction. The algorithm is designed to increase search efficiency by locating and converging on the low energy structures faster than both nGA and PT. Essentially, the strategy corresponds to letting all niches converge before migrating individuals between them and restarting as described by Cantu-Paz and Goldberg [2]. However, while running each niche to convergence worked well for the problem instances chosen by Cantu-Paz and Goldberg, work by Heiler [5] suggest that for protein structure prediction the quality of predicted structures *decrease* when the individuals are locally optimized before the genetic operators are applied. Rather than running to convergence we thus suggest a kind of early stopping, which generate low energy structures without spending too much time on refining suboptimal structures.

2 METHODS

In the traditional niche genetic algorithm (nGA), evolvement of several populations are run in parallel and completely independent from each other. At certain points individuals from one or more niches (islands or demes) are chosen according to some selection strategy and migrated to other niches, where they replace individuals also chosen according to some selection strategy. Usually the selection strategies are based on the fitness values of the individuals such that the best individuals from one niche are migrated to another niche where they replace the worst individuals, as this migration strategy yields the fastest convergence [1].

We propose an iterative niche genetic algorithm (inGA) that performs a type of elitist refinement. Like the traditional niche genetic algorithm multiple populations are initially created and evolved in parallel, but unlike the traditional niche algorithms, individuals do not migrate to other niches. Rather we stop evolvement of all populations after a predefined number of generations, g, and choose the best solution from each of the n niches. The individuals not selected are destroyed while the selected individual are put together in a new population, *pop*. *pop* is then cloned *n* times and the cloned populations are placed on the *n* niches where evolvement of new (and initially identical) populations is then carried out for *g* generations. The procedure of stopping, selecting, cloning and restarting is repeated until the algorithm converges. The pseudo code for the algorithm is given in Algorithm 1

Algorithm 1 Pseudo code for inGA

```
$niches \leftarrow CREATE\_THREADS(n)$
$pop \leftarrow NULL$
while $!DONE()$ do
  for each $n$ in $niches$ do
    if $pop$ equals $NULL$ then
      $population \leftarrow CREATE\_POPULATION()$
    else
      $population \leftarrow CLONE(pop)$
    end if
    $niches[n] \leftarrow start(GA(population, g))$
  end for
  $WAIT\_FOR\_COMPLETION(niches)$
  $pop \leftarrow NULL$
  for each $n$ in niches do
    $pop \leftarrow GET\_BEST\_INDIVIDUAL(niches[n])$
  end for
end while
```

This strategy requires the number of individuals in each niche to be the same as the number of niches, although a different strategy could, of course, also be utilized. In this work we ran 20 parallel niches with 20 individuals in each niche.

We did preliminary experiments to determine how many generations to run the niches in each iteration. We wanted the GA to run just long enough to reach a good solution that captured the best traits of that niche and by analyzing the development of the energy we found that by far the largest improvements happen during the first 100 generations. We thus settled at running 100 generations per iteration. We did try to run the GA for 200 generations, but found that while the initial niche solutions were improved, the final result was not, which is much in keeping with the findings presented in [5].

The selection strategy employed both an elitism strategy and the fitness proportionate selection strategy known as a roulette wheel. The elitism strategy clones and transfers the 10% top scoring individuals unaltered to the next generation thereby ensuring that the best individuals are always kept. However, the 10% best individuals are also allowed to compete in the roulette wheel selection, where each individual is chosen with a probability corresponding to its fitness value. This strategy is chosen over rank selection to ensure a better chance for low scoring individuals to be selected.

Individuals selected by the roulette wheel are subjected to crossover and mutation. A multi-point crossover strategy is used where the number of crossover sites, c, are chosen according to a Gaussian distribution. The c actual crossover sites are chosen at random. The advantage of multi-point crossover over single point crossover is that it eliminates the bias of the end segments that is commonly raised as an issue with the vector representation employed by most genetic algorithms. Also multi-point crossover typically results in bigger alterations of the solutions causing the genetic algorithm to explore very different regions of the search space.

As is often the choice, the mutation rate is set fairly low to a value of 0.001. Mutation is thus not the driving force in the folding process, but is used mainly as a way to introduce new genes into the existing gene pool.

It should be noted that setting the hyper-parameters of genetic algorithms (such as selection and recombination strategies) is an optimization problem in itself. We have looked to the literature for inspiration and carried out numerous experiments with different combinations before settling on the ones described here.

2.1 Encoding

In this work we use a physics based energy potentials, called POISE [8], which requires a full atom model. However, rather than searching the Cartesian space, only dihedral angles, bond angles and bond lengths (referred to as the set of structural variables) are explicitly represented as atom positions can be calculated directly from these using standard matrix operations. A protein is thus encoded as a vector (chromosome) of $S_{structvar}$, where each $S_{structvar}$ represents the structural variable of one amino acid.

One of the problems often encountered during encoding of proteins is the occurrence of clashing atoms. When two atoms come within very close proximity, the laws of physic dictate that the energy will rapidly grow towards infinity and the atoms will be forced apart. When using a full atom model one of two strategies can be utilized; either one can explicitly check and make sure that atoms do not clash or clash are tolerated, but heavily penalized by the energy function, such that solutions with clashing atoms stand little chance of being accepted/selected. The latter works best in conjunction with statistics based energy function where the 'energy' term is usually an artificial pseudo energy made up from many parameters that are non-numerical in nature. With a pure physics based energy potential, the infinitely large increase in energy caused by two clashing atoms will cause overflow on a computer. As we utilize a pure physics based potential, a check for clashing atoms is thus carried out, before a solution is evaluated and only clash-free structures are accepted.

The protein is encoded sequentially and for every new residue added we check whether the $S_{structvar}$ chosen causes any atoms of the new amino acid to clash with atoms in the residues that have already been added. If a clash occur a new $S_{structvar}$ for the amino acid is chosen. If the problem with atom clash has not been resolved after 20 different $S_{structvar}$ have been tried for the amino acid, we backtrack and choose a new $S_{structvar}$ for the previous residue. Although the theoretical running time for this strategy is $O(2^n)$ the running time was not found to be an issue in practice.

2.2 Move set

The move set is defined as the set of possible combinations of bond lengths, angles and dihedral angles for each amino acid. Theoretically, the move set is unrestricted, but in practice we know that bond lengths and angles vary very little and dihedral angles are heavily biased towards certain areas of the dihedral angle space. Here, we thus choose bond angles and bond lengths for amino acids randomly from within a small interval (up to ± 0.1) of the optimal angles and lengths as defined in the AMBER 99 parameters.

The dihedral angles space is likewise restricted. In Fonseca and Helles [4] a probability distribution is predicted for each amino acid in a sequence by considering the neighboring amino acids. This probability distribution is used here such that the dihedral angles are chosen from this sequence-dependent distribution thereby maximizing the probability of sampling a realistic area of the dihedral angle space.

2.3 Energy function

Generally speaking, energy functions can be divided into two categories: physics based energy function and statistics based energy functions. The energy function used here, called POISE, is a purely physics based potential, described in more details in [8]. It combines the AMBER force field [12]:

$$\begin{aligned} E_{\text{protein}} = & \sum_{\text{i}}^{bonds} K_b(b_i - b_0)^2 + \sum_{\text{i}}^{angles} K_\theta(\theta_i - \theta_0)^2 \\ & + \sum_{\text{i}}^{dihedrals} k_x[1 + \cos(n\phi - \gamma)] + \\ & + \sum_{i}^{N}\sum_{i<j}^{N}\left(\frac{q_i q_j}{r_{ij}} + 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]\right) \end{aligned} \tag{1}$$

with a Generalized Born component:

$$V_{\text{GB}} = -\frac{1}{2}\left(\frac{1}{\varepsilon_p} - \frac{1}{\varepsilon_w}\right)\sum_i\sum_{i<j}\frac{q_i q_j}{f_{ij}^{GB}(r_{ij})} \tag{2}$$

$$f_{\text{ij}}^{GB}(r_{ij}) = \left[r_{ij}^2 + R_i R_j exp\left(\frac{-r_{ij(2)}}{4R_i R_j}\right)\right]^{\frac{1}{2}}$$

	Category	Residues
1C75	α	71
1NKD	α	59
1YK4	β	52
2O9S	β	67
1EJG	αβ	46
1IQZ	αβ	81

Table 1: Test proteins

and a hydrophobic mean force potential:

$$V_{\mathrm{HMFP}} = \sum_{i \in SA_i > A_c}^{N_c} tanh(SA_i) \sum_{j \in SA_j > A_c, j \neq i}^{N_c} tanh(SA_j)$$
$$x \sum_{k=1}^{3} h_k exp\left(-\left[\frac{r_{ij} - c_k}{w_k}\right]^2\right) \quad (3)$$

We refer to [8] for an explanation of the parameters.

The potential considers interatomic energies between all pairs of atoms and the time complexity of calculating the potential is thus quadratic ($O(n^2)$). Fortunately, the vast majority of atoms in a protein are simply too far apart to exert any power on each other and by simply omitting calculations of these interactions, the function is implemented such that the potential is calculated in linear time.

It should be noted, that in this current work we are primarily concerned with testing the performance of different parallel meta-heuristics. We have chosen the POISE energy function because it gives a realistic impression of the very rugged fitness landscape that the algorithms have to navigate around

2.4 Test proteins

A standardized, diverse set of proteins that is guaranteed to provide an adequate representation of protein structures does unfortunately not exist. The test set used here is constructed such that 2 proteins from each of the categories α, β and αβ have been picked from the PDB Select 25% [2]. Only small to medium sized proteins have been included. The smallest protein includes 46 amino acids and the largest protein includes 81 amino acids (Table 1).

2.5 Benchmarking algorithms

In order to evaluate the performance of inGA we implemented both the traditional nGA and the parallel version of SA, called parallel tempering (also known as the

[2]http://bioinfo.tg.fh-giessen.de/pdbselect/recent.pdb_select25

Replica Exchange Monte Carlo algorithm). Parallel algorithms have numerous times been reported to outperform the sequential algorithms, and here we thus focus only on parallel algorithms [3].

The nGA use the same GA to evolve populations as in inGA, and like for inGA we also used 20 parallel niches in nGA. The convergence rate of a nGA is strongly affected by the migration scheme [1]. Migrating and replacing only a few randomly chosen individuals leads to very slow convergence whereas migrating the best and replacing the worst leads to the fastest convergence. To make the comparison with inGA, which is highly elitist, fair, we employed a rather strong selection scheme such that every 100 generation we chose the 50% best individuals from each population, cloned them and migrated them to another niche where it replaced the 50% worst individuals.

The PT algorithm is implemented such that it utilizes the same encoding strategy, energy function and move set as described above in order to ensure comparability. The lowest and highest temperatures were determined in the way proposed in [11], such that

$$c_{highest} = -\delta E_{max}/ln(P^A(\delta E_{max})) \tag{4}$$

and

$$c_{lowest} = -\delta E_{min}/ln(P^A(\delta E_{min})) \tag{5}$$

Initial experiments measuring differences in the energy, E, between neighboring structures were run to determine the values of E_{max} and E_{min}. $P^A(\delta E_{max})$ and $P^A(\delta E_{min})$ were set to 0.95 and 0.05 respectively. This resulted in $c_{highest}$ = 3800 and c_{lowest} = 10 with temperatures of the different replicas spaced according to:

$$temp\_replica_{i+1} = 10 * i * 2 + temp\_replica_i \tag{6}$$

The observed average probability of accepting a swapping move between neighboring replicas was roughly 20% in accordance with [7], but with swapping of course occurring much more frequently between replicas running at high temperatures and much less frequently between replicas running at low temperatures.

In each time step every replica goes through N moves for a N-residue long protein. A move consists of randomly selecting an amino acid and picking a new $S_{structvar}$ for that amino acid. As is typical for the simulated annealing approach a move from structure s to some neighbor s' is accepted with the following probabilities:

$$P(accept) = \begin{cases} exp^{-(E_{s'}-E_s)/K_bT} & E_{s'} \geq E_s \\ 1 & E_{s'} < E_s \end{cases} \tag{7}$$

where E is the energy of the structure and T is the temperature.

[3] Although the results are not reported here, experiments with both the standard non-parallel genetic algorithm and simulated annealing was carried out as well, and they did indeed perform worse than the parallel versions

	nGA	inGA	PT
1C75	319	**287**	513
1NKD	113	**-14**	200
1YK4	216	**142**	275
2O9S	385	**223**	875
1EJG	4	**-7**	30
1IQZ	414	**348**	665

Table 2: Early results of the three algorithms. Energies are calculated with the POISE potential. Lower energies are better

We ran 20 parallel simulations. The probability of accepting a swap between to replicas, i and j, was given by:

$$P(i \leftrightarrow j) = min\{1, exp^{[-(\beta_i - \beta_j)(E_j - E_i)]}\} \quad (8)$$

where β is the inverse temperature $\beta = 1/k_B T$ and $\beta_i > \beta_j$. Defining the probability of swapping replicas such that it decreases exponentially as the gap between temperature increase is usually employed in PT [3] and also the reason why it was chosen here.

3 RESULTS AND DISCUSSION

Early results from experimentation with the three different parallelization schemes on the test proteins, shown in Table 2, look very promising with inGA quickly and consistently locating structures of lower energy than both nGA and PT. Please note that we have not calculated RMSD of the final structures, because as such the search algorithms are all oblivious to the concept of a native structures. They seek merely to minimize energy as specified by the POISE potential and in this experiment we are only interested in determining how efficient the different algorithms are in finding low energy structures in the highly rugged energy landscape associated with protein structure prediction energy functions. A different energy function would most likely lead to different (either better or worse) quality of the final structures in terms of RMSD to the native structure, but the differences in how well the algorithms perform with respect to each other would (expectedly) remain the same.

Given unlimited time, all meta-heuristics would probably find the same low energy structures. Unfortunately, time is usually not unlimited in practice and designing search algorithms that increases search efficiency such that we can obtain better results faster becomes important. Parallelization has in itself increased search efficiency, but from the results presented here it is evident that how the algorithms are parallelized can also have a profound impact on how efficiently the algorithms travel the energy surface in their search for the global minima.

The solution space for a given protein sequence is infinitely big and the key to success for a meta-heuristics is usually a good balance between exploration and exploitation. Minima should be explored thoroughly while still allowing the algorithm to move relatively freely across energy barriers. PT, nGA and inGA all differ from each other in this exploration-exploitation balance.

In PT the balance between exploration and exploitation is kept by running parallel simulations at different temperatures. The advantage of PT is that it can be run for exactly as long as time permits, because while it may settle at a minima, it does not really converge but rather keep exploring for a preset number of iterations or until it is interrupted. As such the PT algorithm enjoys the same theoretical guarantee of finding the global minima as the simulated annealing algorithm. However, while parallel tempering reaches low energy structures faster than sequential Monte Carlo simulations [3], the number of replicas used depend not so much on available processors, but on what makes sense in order to maintain proper communication between the different replicas. In other words, there appears to be an upper limit on what we can expect to gain in performance that depend on the problem and not on CPU power. For proteins of the length used here, 20 replicas ensure appropriate communication across temperatures, and more replicas would thus only increase the level of communication thereby setting off the exploration-exploitation balance, which would not be desirable. Of course, for larger proteins where the energy span between different structures is likely to be greater than for the proteins used here, more replicas would most likely be required to ensure proper communication.

One of the reasons why PT does not reach the low energy structures as fast as the genetic algorithms is that although many replicas are run at the same time they do not exchange information between replicas. If a good solution is encountered at one temperature it may be exploited by swapping it to a lower temperature, but it does not share its favorable characteristics with any of the other replicas. Parallel tempering would most likely reach the same results as achieved by inGA, but we postulate that because of the lack of information sharing it can generally be assumed to take much longer.

The genetic algorithms on the other hand have a high degree of information sharing via their crossover operator, which explains why the genetic algorithms reach the lower energy structures much quicker. The migration scheme we have used here for nGA is fairly aggressive to ensure faster convergence that would be comparable with inGA. From the results it is evident that while nGA finds lower energy structures than PT it does not reach structures with energies as low as inGA. It should also be noted that despite the aggressive migration strategy, nGA does not fully converge within 20 iterations for any of the test proteins, although signs of convergence is beginning to show. We did initially experiment with a less aggressive migration scheme (that migrated only the best individual), but energies were significantly worse after the 20 iterations than with the chosen migration scheme and it did of course not come near convergence within 20 iterations.

An issues with inGA is that it may simply converge prematurely. The iterative strategy of inGA is highly elitist and with 20 niches it usually converges fully within 20 iterations for the small to medium sized proteins used here. An elitist strategy (always picking the best) favors exploitation heavily and will normally only work well in smooth energy landscapes. The energy landscape of proteins is obviously anything but smooth, but interestingly a balance with exploration does nevertheless appear to be maintained in inGA by the niche approach. Exploration can thus be controlled by simply adding more or less niches. This is indeed a nice feature, since better performance can then be brought to depend more on available CPUs rather than on available time. Obviously, adding more niches would most likely require more iterations to fully converge, but the number of iterations required to converge would expectedly grow much slower for inGA than for nGA thereby making the difference in performance between inGA and nGA greater as the number of niches increase.

4 CONCLUSIONS

We presented an iterative variant of the parallel niche genetic algorithm for protein structure prediction. Early results show that the algorithm finds significantly lower energy structures than both the traditional niche genetic algorithm and the parallel tempering algorithm within comparable time. The algorithm converges quickly and the exploration-exploitation balance can be controlled with the number of niches included, which means that search efficiency can be expected to scale nicely with the number of available CPUs.

References

[1] E. Alba. *Parallel Metaheuristics.* Wiley, 2005.

[2] E. Cant-Paz and D. E. Goldberg. Modeling idealized bounding cases of parallel genetic algorithms. In *In*, pages 353–361. Morgan Kaufmann Publishers, 1996.

[3] D. J. Earlab and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem*, 7:3910, 2005.

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

[5] M. Heiler. Massively parallel gas for protein structure, 1998.

[6] G. Helles. A comparative study of the reported performance of *Ab Initio* protein structure prediction algorithms. *J. R. Soc. Interface*, 5:387396, 2008.

[7] A. Kone and D. A. Kofke. Selection of temperature intervals for parallel-tempering simulations. *J. Chem. Phys.*, 122:206101, 2005.

[8] M. S. Lin, N. Lux Fawzi, and T. Head-Gordon. Hydrophobic potential of mean force as a solvation function for protein structure prediction. *Structure*, 15:727–740, 2007.

[9] M. T. Oakley, D. Barthel, Y. Bykov, J. M. Garibaldi, E. K. Burke, N. Krasnogor, and J. D. Hirst. Search strategies in structural bioinformatics. *Curr. Prot.Pep. Sci.*, 9:260274, 2008.

[10] G. N. Ramachandran and V. Sasisekharan. Conformations of polypeptides and proteins. *Adv. Prot. Chem.*, 23:283–437, 1968.

[11] H. Sanvicente-Snchez and J. Frausto-Sols. A method to establish the cooling scheme in simulated annealing like algorithms. *LNCS*, 3945:755–763, 2004.

[12] Robert H. Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Phys. rev. let.*, 57:2607–2609, 1986.