`SaDSaT`
a Knowledge Based Potential for Protein Folding

Diplomarbeit

zur Erlangung des akademischen Grades

Magister rerum naturalium

an der Fakultät für Naturwissenschaften und Mathematik
der Universität Wien

Vorgelegt von

Stephan Bernhart

im Oktober 2003

Diese Arbeit sei meinen Eltern und Brigitte gewidmet.

> “Some questions should not be asked.
> However, someone always does.
> ’How does it work?”’

T. Pratchett, I. Stewart, J. Cohen, The Science of Discworld, page 15.

Diese Arbeit entstand von Frühling 2001 bis Herbst 2003 am Institut für theoretische Biochemie und molekulare Strukturbiologie der Universität Wien.

Ich möchte meinem Betreuer Prof. Dr. Peter F Stadler für seine Geduld und Unterstützung meinen herzlichsten Dank aussprechen. Auch meinem informellen Zweitbetreuer, Dr. Ivo L Hofacker, gebührt besonderer Dank für sein geduldiges Ertragen meiner, hoffentlich zeitweiligen, Begriffstutzigkeit sowie für seine Hilfestellungen, die er zu jeder Unzeit für mich hatte.

Dank auch an Prof. Dr. Arnold Neumaier vom Institut für Mathematik für die Hilfestellung bei der Lösung eines mathematischen Problems.

Ich möchte mich auch bei der Belegschaft des tbi für die wunderschöne Zeit hier bedanken. Speziell bei meinen Zimmerkoleginnen Uli und Stefanie, bei seiner Administranz MTW, bei seiner Instanz Roman, bei “Regex” Xtof und bei “Professor” Svrci.

Ich danke meiner Familie für ihre Unterstützung und dafür, dass sie an mich geglaubt haben. Meine Eltern möchte ich speziell für ihre Sponsorentätigkeit und ihre Geduld hervorheben, meinen Bruder dafür, dass er mir als Beispiel gedient hat.

All meine Freunde müssen hier auch Erwähnung finden, speziell Answer, Klaus, Nina, Pauli, Rainer und Stefan, auch dafür das sie mir Ausgleich zum manchmal stressigen Studienalltag gebracht haben.

Dank all meinen StudienkolegInnen, Andrea, Beate, Corinna, Jiří, Klaus, Sigrid und allen anderen, die mich durchs Studium getragen haben.

Und vor allem Dank an Brigitte für den grünen Himmel und den Jackpot.

Abstract

The overwhelming amount of known protein sequences from large scale sequencing projects can sometimes obscure the relevant information. It is easy to translate the DNA sequence to the corresponding protein chain, but today still impossible to gain access to the structure from this sequence information. But the amino acid sequence separate from its $3d$ context is often meaningless. Nevertheless sequence homologies and alignment studies are useful tools. A detailed mapping of the hyper-astronomic sequence space of proteins would be a hopeless task if the number of distinct stable folds would not be restricted. Such a map would be extremely useful for evolutionary studies as well as *de novo* protein design.

Babajide and co-workers revealed that knowledge based potential are suitable means to perform an analysis of protein space, targeting inverse folding. The basic assumptions of knowledge based potentials are that proteins exist in the energetic ground state and that the inverse Boltzmann law is valid.

Empirical potentials are extracted from databases of known structures. They mainly vary in the definition of residue interaction. Usually, they use distance criterions to define contacts. Avoiding the arbitrariness of a binned distance, Alexander Tropsha introduced a "statistical geometry" approach. The polypeptide chain is unambiguously partitioned by means of Delaunay tessellation. This yields a cluster of tightly packed, irregular tetrahedrons having an amino acid at each corner. Tropsha thus defines a potential not including any distance information. However, Manfred Sippl showed that potentials using distance information yield very good results. He used the pairwise distances of amino acids, split in euclidean as well as sequence distance classes, to define a knowledge based potential.

The idea of this work is to try and combine these two ways of defining a potential, thus attempting to increase the performance of tessellation based potentials in inverse folding experiments as well as in other applications. Furthermore, the description of protein-solvent interactions was improved by introducing an explicit shell of water.

This work describes the implementation of this combined potential and a few applications to inverse folding computer experiments. A distinct improvement compared to previous tessellation based potentials could be obtained and possibilities to further improve the discrimination power are pointed out.

Zusammenfassung

Eine überwältigende Flut an Sequenzinformation aus groß angelegten Sequenzierexperimenten erschwert immer mehr den Blick auf relevante Informationen. Man kann zwar DNA Sequenzen leicht in die entsprechende Proteinsequenz übersetzen, jedoch ist diese ohne die dazugehörige $3d$ Struktur oft nutzlos. Ein detailliertes Mapping des astronomisch großen Proteinsequenzraumes wäre völlig aussichtslos, wäre nicht die Zahl der stabilen Strukturen sehr begrenzt. Eine solche Kartierung wäre von großem Nutzen für evolutionäre Studien und das *de novo* Design von Proteinen.
Babajide *et al.* konnten zeigen, dass empirische Potentiale zur Erforschung des Proteinraums mittels inverser Faltung geeignet sind. Die diesen Potentialen zugrunde liegenden Annahmen sind, dass das Protein im energetischen Grundzustand vorliegt, und dass das inverse Boltzmann Gesetz gilt.

Empirische Potentiale werden aus Datenbanken struktureller Informationen extrahiert. Sie unterscheiden sich normalerweise in der Definition der berücksichtigten Wechselwirkungen. Meistens werden Distanzkriterien dazu benutzt, Kontakte zu definieren. Alexander Tropsha konnte die Willkür eines gewählten Abstandes umgehen, indem er Methoden der statistischen Geometrie einführte. Die Proteinkette wird hierzu der Delaunay Tessellation unterzogen. Dies führt zu einem Cluster von dicht gepackten Tetraedern mit einer Aminosäure an jeder Ecke. Tropsha definiert so ein Potential, das keinerlei Distanzinformation verwendet. Manfred Sippl konnte aber zeigen, dass ein Potential das solche Distanzinformationen verwendet sehr gute Resultate liefert. Er definierte sein empirisches Potential aus den paarweisen Distanzen der einzelnen Aminosäuren und teilte diese Distanzen dazu in Sequenz- und Euklidische Abstandsklassen ein.
Die Idee dieser Arbeit ist die Verschmelzung dieser zwei Ansätze zu einem einzigen empirischen Potential, um die Qualität von auf Delaunay Tessellation basierenden Potentialen zu erhöhen. Darüberhinaus wird die Beschreibung der Protein-Lösungsmittel Wechselwirkung dadurch verbessert, dass explizit eine Wasserschale eingeführt wird.
Diese Arbeit beschreibt die erfolgreiche Verschmelzung der zwei Potentiale und deren Anwendung in Simulationen von inversen Proteinfaltungen. Die Computerexperimente zeigen eindrucksvoll eine deutliche Verbesserung gegenüber Ergebnissen früherer Tessellationsbasierter Potentiale. Aufgrund der vorliegenden Ergebnisse werden auch weitere Verbesserungsvorschläge gemacht, die in dieser Arbeit nicht oder nicht mehr durchgeführt werden konnten.

Contents

1 Introduction 1

2 Theory 4

2.1 Molecular Force Fields or Knowledge Based Potentials? 4

2.2 Force Fields 5

2.3 Knowledge Based Potentials 9

2.3.1 The Inverse Boltzmann Law 10

2.3.2 Approximations used in Knowledge Based Potentials . 11

2.3.3 Profiling Potentials 11

2.3.4 Contact Potentials 14

2.3.5 Lapedes' Neural Network `NN` Potential 16

2.3.6 Atom-Atom Potentials 17

2.3.7 Sippl's `PROSAII` Potential 18

2.3.8 Problems of Knowledge Based Potentials 19

2.4 Energy and Z-score 20

2.5 Delaunay Tessellation 22

2.5.1 The Voronoi Diagram 22

2.5.2 Delaunay Triangulation 23

2.5.3 The `qhull` Algorithm 24

2.5.4 Empirical Protein Potentials from Delaunay Tessellation 27

2.5.5 Four Body Contact Potentials 27

3 The `SaDSaT` Potential **29**

3.1 Overview of Innovations 29

3.1.1 Distance Classes 29

3.1.2 Surface Terms 31

3.2 Pairs 31

3.3 Triangles 32

3.3.1 Principle of Maximum Entropy 32

3.3.2 Application 33

3.4 Tetrahedrons 35
  3.4.1 Computation using Triangles 36
  3.4.2 Tetrahedrons dependent on Pairs 37
3.5 Surface 39
  3.5.1 Water Shell 39
  3.5.2 Expected Frequencies of Surface Contacts 43
3.6 Data Base 45
3.7 Sparse Data 46

4 Computational Results 50
4.1 Calibrating the Potential 50
  4.1.1 Calibrating the Water Grid 50
  4.1.2 Calibrating the Scale of the Surface Term 52
  4.1.3 Calibrating the Sequence and Euclidean Distance Classes 53
4.2 Performance of the Potential 57
  4.2.1 Z-scores 57
  4.2.2 Decoy Sets 58
  4.2.3 Stability Changes 63
4.3 Inverse Folding 65
4.4 Computational Effort 71
4.5 Iterative Proportional Fitting 73
  4.5.1 Results using IPF 73

5 Discussion 80

References 83

A Amino Acids i

B Binomial Coefficients vi

C Library vii

D Programs **xiv**
D.1 `SaDSaTstat` xiv
D.2 `CompLogLike` xvii
D.3 `SaDSaT` xx
D.4 `SaDSaTinv` xxiii

1 Introduction

Proteins are the machinery of life. As enzymes, they catalyze almost any biochemical reaction in the cell with very high accuracy and velocity, leading to myriads of different products as simple as methane or as complex as alkaloids. They play the key role in light harvesting during photosynthesis, converting electromagnetic energy to a potential gradient. They form the cytoskeleton and are the main components of hair or the silk of spiders, which have properties that can not easily be reproduced by other materials.

This versatility is achieved by hetero-polymers composed of 20 different monomers, the amino acids. All proteins share the same basic outlay, but the exact sequence of amino acids determines the 3-dimensional structure of the peptide chain. This 3-dimensional structure is responsible for the different ways proteins interact with their environment and with each other.

The question of how exactly the sequence of amino acids (primary structure) determines the 3-dimensional (tertiary) structure is one of the most important in contemporary bioscience. It was estimated in 1991 that for every tertiary structure measured by NMR or X-ray crystallography there were 50 primary structures obtained [12]. With the extensive "genome projects" in recent years, it is safe to say that this proportion is even more in favor of the primary structures today. This leads to a big accumulation of data the exact meaning of which has yet to be uncovered.

In contrast to nucleic acids, the other class of bio-polymers in cells, where at least the secondary structure can be predicted reliably [36, 69], for proteins even their secondary structures are hard to predict. This is mainly attributed to the more or less unspecific hydrophobic interactions involved in protein folding, which are hardly characterized or measured. Furthermore, the secondary structure of proteins is defined locally and does not contain long distance interactions. Therefore, it is not as meaningful as RNA secondary structure. Thus, we have to deal with tertiary structure directly. In principle, it is assumed that the native fold of proteins corresponds to the minimum of its free energy function $W(S)$. If this function is known, any sequence S can be assigned to its fold $\psi(S)$. Alas, the number of terms contributing to this function is enormous, depending on the sequence of amino acids as well as on the natural environment (pH, temperature, ionic strength, solvent type etc.). Thus finding the minimum of the free energy function is a hard optimization problem even assuming all the necessary terms are known.

Different ways to solve this problem are used today. The best ones try to find a protein with known structure and similar sequence and then align the new sequence to this structure. We know that although there is an astronomic number of possible amino acid sequences (20^n sequences of chain length n) [52], the number of tertiary structures occurring in nature seems to be limited, with 90% of the native sequences sharing only 930 stable folds [29] and a total number between 4000 and 8000 folds estimated [1, 29, 64]. We also know that there are neutral networks in the sequence space [4, 37], which can lead to proteins with a high structural but no perceivable sequential homology. This can of course make it impossible to recognize the correct fold using the sequence alignment techniques only.

Thus, other ways of finding the tertiary structure of proteins are needed.

Another approach uses known tertiary structures of proteins to try and find something like "emergent rules". By defining different types of interactions, checking their frequency in a data base and assigning an energy to them, these so called "knowledge based" approaches design a potential using statistical mechanics. These knowledge based potentials, because they are using coarse grained features of the structure, are potentials of free energy. Thus they can be used to find the corresponding structure for a sequence. However, they can not easily be used for sequence-structure alignment, as their nature makes it almost impossible to use dynamic programming algorithms.

Another interesting question is the exact opposite one: which sequence will fold into a given tertiary structure? The versatility of proteins is shown in many different applications, e.g. catalysts in organic synthesis or cures for diseases. To be able to tailor proteins for any use, to change e.g. the substrate specificity of an enzyme or the paratope of an antibody, it is preferable to know *a priori* whether a mutant protein will retain the desired structure or not, since the 3-dimensional structure determines the function of the protein. Knowledge based potentials should be able to fulfill this function, making it possible to save time and resources when designing mutations or even *de novo* proteins.

Other questions not answered to any extent as of now concern the theoretical background of proteins. How are they able to fold so quickly? How do they interact with each other and with the other molecules of their native environment? While there is a lot of work invested in answering these questions (see e.g. [53, 60, 63]), they are still far from solved. Thus, new computational methods and models to help understanding proteins are needed.

In this work, we try to improve a knowledge based potential primarily to be used for inverse folding of proteins. There is a multitude of different ways to create knowledge based potentials, as many people have been working on this field for well over ten years. We use elements of the works of Sippl and Tropsha [47, 49] to develop a new potential of increased quality. To make the results of this work available for other scientists, we develop a library of functions as well as stand-alone programs called `SaDSaT` accessible over the Internet.

2 Theory

2.1 Molecular Force Fields or Knowledge Based Potentials?

The energy of a macromolecular system is a function of its conformational variables (e.g. Cartesian coordinates or Bond lengths and -angles) plus its interaction energy with the surrounding solvent. The derivation of the energy from the conformational variables leads to the force field of the molecule. Generally we assume that a protein sequence $S = (s_1, ..., s_n)$ of n amino acids

$$s_i \in \{\mathbf{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y}\}$$

is related with its structure ψ, represented by the coordinates representing this amino acids $\psi = (x_1, ..., x_n)$ via the potential function
$V(S, \psi)$:

$$\psi = \arg_\psi \min V(S, \psi)$$

There are at least two different approaches to designing a molecular force field:
On the one hand, semi-empirical approaches consider macromolecular systems as a summation of the forces observed for monomers. These force fields use quantum mechanic calculations as well as data obtained by thermodynamic or spectroscopic measurements on small molecules to generate a potential for macromolecules.
Knowledge based potentials, on the other hand, try to extract information out of data bases of macromolecular structures, assuming that the force fields of macromolecules are of too great complexity and hence the only reliable source of information are the macromolecules themselves.

2.2 Force Fields

The principles of force fields (also known as molecular mechanics) are based upon Newtonian mechanics. The basic idea is that bond lengths, valence and torsional angles have “natural” values depending on the involved atoms and that molecules try to adjust their geometries to adopt these values as closely as possible. Additionally, steric and electrostatic interactions, mainly represented by van der Waals and Coulomb forces, are included in the so-called potential. Basic ideas for these calculations go back to the work of Andrews in 1930 [2], the first serious applications of force field methods date back to 1946 [33, 24].

The basis of molecular mechanics derives from the accuracy of the Born-Oppenheimer approximation [9], which describes the motion of the nuclei of molecules on a so-called “potential surface”, caused by the electronic structure. The Born-Oppenheimer approximation works because electrons respond almost instantaneously to changes in nuclear positions.

A typical force field contains a set of several potential functions which themselves contain adjustable parameters. These parameters are optimized to obtain the best fit to experimental values, such as geometries, conformational energies and spectroscopic properties. It is important to realize that force fields are usually parameterized for a limited set of molecular properties and a specific set of molecules. If certain parameters are not experimentally available, quantum mechanical calculations of representative fragments are used to obtain the desired values.

Many of the molecular modeling force fields in use today consist of relatively simple four component picture of intra- and intermolecular forces within the system (Figure 1.

$$E_{total} = E_{bond} + E_{angle} + E_{torsion} + E_{non-bonding}$$

In the simplest approach the energy terms are in detail :

Bond - Energy:
The energy between two bonded atoms. It increases when the bond is compressed or stretched. The potential is described by an equation based on Hooke's law for springs.

$$E_{bond} = \sum_{bonds} k_b (r - r_0)^2$$

whereby k_b is the force constant, r is the actual bond length and r_0 the equilibrium length. The equilibrium length as well as the bond constant have to be acquired for each pair of atom types.

Angle Energy:
The energy of the binding angle of three atoms. It increases if bond angles deviate from their reference positions. Again the approximation is harmonic and uses Hooke's law.

$$E_{angle} = \sum_{angles} k_\theta (\theta - \theta_0)^2$$

k_θ controls the stiffness of the angle, θ is the current bond angle and θ_0

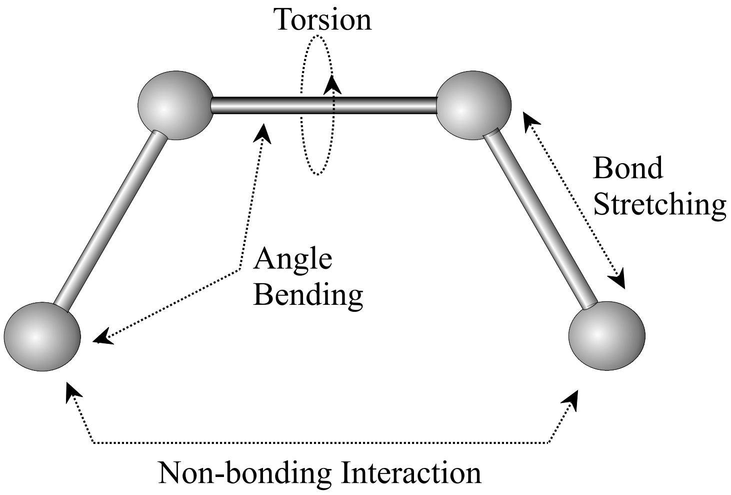


Figure 1: Four component picture of inter- and intra-molecular forces, adopted from [31]

the equilibrium angle. Both, the force and equilibrium constant have to be estimated for each triple of atom types.

Torsion Energy:
The energy of the torsional angle of four atoms. Intra-molecular rotations (around torsions or dihedrals) require energy as well:

$$E_{torsion} = \sum_{torsions} \frac{V_n}{2}(1 + \cos(n\omega - \gamma))$$

V_n controls the amplitude of this periodic function, n is the multiplicity, and γ the so-called phase factor, shifts the entire curve along the rotation angle axis ω. Again the parameters V_n, n and γ for all combinations of four atoms have to be determined.

Non-bonding Energy:
The simplest potential for non-bonding interactions includes two terms, a Van der Waals and a Coulomb term.

$$E_{non-bonding} = \underbrace{\sum_i \sum_{j>i} \left(\frac{A_{ij}}{r_{ij}^6} - \frac{B_{ij}}{r_{ij}^{12}} \right)}_{\text{Van der Waals}} + \underbrace{\sum_i \sum_{j>i} \frac{q_i q_j}{r_{ij}}}_{Coulomb}$$

The Van der Waals term accounts for the attraction and the Coulomb term for electrostatic interaction. The shown approximation for the van der Waals energy is of the Lennard-Jones 6-12 potential type.

These simple terms mentioned above can be expanded to adjust the potentials better to experimental results (e.g using Morse potential for bonds, Taylor expansions with higher terms, cross-terms between the potentials), but with the disadvantage of higher computational effort. That is the reason why bio-molecular force fields, e.g. CHARMM [14] or AMBER [45] usually can not afford to include these refinement terms for the bond, angle and torsion potential.

Sometimes force fields include additional potential terms for specific interactions, such as hydrogen bonding, polarization or dipole-dipole interactions. The most critical term, for biomolecules also, are the non-bonded, mainly Van der Waals and Coulomb, interactions.
First the number of non-bonded interactions in a molecule grows as $\frac{n(n-1)}{2}$, where n is the number of atoms in the molecule. Choosing a complex term for the non-bonding interactions results in a tremendous increase in computational effort. Second, this non-bonded interaction term must include the solvation effects, because biomolecules usually exist in an aqueous environment. This solvation has a major influence on the electrostatic forces. It is included either by explicitly including solvent molecules and counter ions, or implicitly by representing the solvent as a dielectric continuum and adding a corresponding term to the force field.

Molecular mechanics plays a crucial role in structure determination by NMR, where it is used to refine model structures subject to certain constraints. It is used to minimize the energies of NMR as well as X-ray derived empirical structures. But there are certain problems if Molecular Mechanics force fields are to be applied to protein structure prediction. Firstly, it is very hard to find adequate potentials, for proteins as well as for RNA. Another problem is the high level of detail of these forcefields, leading to exceptionally high computational costs. These costs make it impossible to sample the very big and rugged energy landscapes of proteins.
Furthermore, molecular mechanics force fields do not include the entropy of the system, so they cannot be used to compute a free energy directly. To be able to get entropic contributions, molecular dynamics is used. So the computational problem is not only to find a minimum of the energy function, but also to compute a trajectory sufficiently long for simulating entropic effects. Dependent on the protein, this trajectory may have to be ms or s long, which can, if ever, only be achieved at astronomic computational costs. So, while it has been shown to be possible to use molecular mechanics

force fields for discriminating native from mis-folded proteins [40], there are much faster ways to achieve this.

There are a lot of parameters influencing the final result of the folding. The question of the best parameter set for any given task is yet to be solved, and it is even possible that there is none for protein folding, i. e. that different protein folding problems will need different parameters. Basically, to find the global minimum of said energy landscape it is necessary to start at a reasonably close spot - like when minimizing X-ray derived structures. It is not yet established whether molecular mechanics is accurate enough to be of use for *ab initio* folding in principle [56]. Bryngelson [15] states that to be able to accurately predict protein structures requires monomer-monomer interaction energies accurate to within 5% to 15%. Ponder *et al.* [46] state that while there are no force fields with sufficient accuracy as of yet, there are many ideas of how to improve the force fields, including electronic polarizability and simplifying solvent interactions with continuum ideas.

2.3 Knowledge Based Potentials

In contrast to the analytic approach of molecular mechanics force fields, which can be deduced from first principles, knowledge based potentials are a more coarse grained approach which uses free energy. They are contributed to Tanaka [57] and developed by Eisenberg [12] and Miyazawa [41]. Knowledge based Potentials are motivated by the observation that the frequency of a certain observed structural descriptor is related to its free energy by statistical mechanics. They assign an energy for a certain event using a likelihood. This likelihood of finding any particular event is extracted from a data base of known structures, a procedure akin to *data mining*. The increase of information is measured by the log-likelihood ratio of the Bayesian events [5], the relation of prior expected events and observed occurrences. The log-likelihood is a kind of measure for the “surprise” provided by the data base. The physical interpretation of this probability function is founded on statistical mechanics. Based on the assumption that the native protein is at the thermodynamic equilibrium, the low energy elements must occur more frequently than high energy elements. This dependence of occurrence on free energy follows a Boltzmann statistic:

$$f_{occ} \sim \exp\left(-\frac{F}{RT}\right)$$

with R the gas constant, F the free energy and T the conformational temperature.

If the frequency of occurrence can be estimated, it is in principle possible to gain access to the putative energy of a certain fold $\psi(S)$. This interpretation of knowledge based potentials, introduced by Manfred Sippl [50], is the basis for most of contemporary potentials of mean force.

2.3.1 Statistical Thermodynamics of Proteins or the Inverse Boltzmann Law

The so called "*folding postulate*" states, that "*In equilibrium the native state of a protein-solvent system corresponds to the global minimum of free energy*". This was demonstrated in the pioneer study performed by Anfinsen in 1973 [3]. He was able to show that by reducing and re-oxidating disulfide bonds in ribonuclease no loss of function occurs, i.e. that folding is a reversible process. Over the past years many different approaches to potentials of mean force have been made. The various potential functions are distinct in the definition as well as in the order of interaction. Therefore different "resolutions" are used to define the energy functions. The spectrum reaches from an atomic resolution mode (Sippl [48]) to simplified **HP**-patterns (Crippen [21], see Chapter 2.3.4), and a lot in between.

Munson *et al.* [42] were able to show that increasing the order of interaction improves the statistical significance of the terms. Starting with a highly significant one body term that counts for the exposures of the residue and continuing with a pair potential term that contributes to amino acid preferences (e.g. hydrophobic-hydrophobic interactions) independent of the burial status, one can clearly identify that multi-body interactions participate to a major extent to the overall potential function.

2.3.2 Approximations used in Knowledge Based Potentials

While Boltzmann statistics applies to a thermodynamical ensemble of structures, knowledge based potentials replace these structures of the thermodynamic equilibrium with structures represented in a data base. Simplified, a knowledge based potential uses some kind of structural descriptor D and assigns an energy e_D to it. The free energy E of the molecule is simply the sum of all the energies for all the descriptors occurring in the molecule:

$$E_{S,\psi} = \sum_D e_D \tag{1}$$

Thus it is assumed that these energies are independent of each other, because otherwise they would not be additive. While this is not true, as there are relations between the energies, of course, it is necessary to use this approximation to be able to compute energies at a reasonable computational cost.

2.3.3 Profiling Potentials

In the simplest application of knowledge based potentials, Eisenberg and coworkers [12] decided to "translate" the $3d$-structures to a $1d$-string, using three parameters:

1. The total side-chain area being covered by any other protein atoms
2. The fraction of side-chain area being covered by polar atoms or water molecules
3. The local secondary structure

First, the "area buried" and "fraction polar" criteria are used to divide the environment in 6 classes (see Figure 2). These 6 classes are further decided

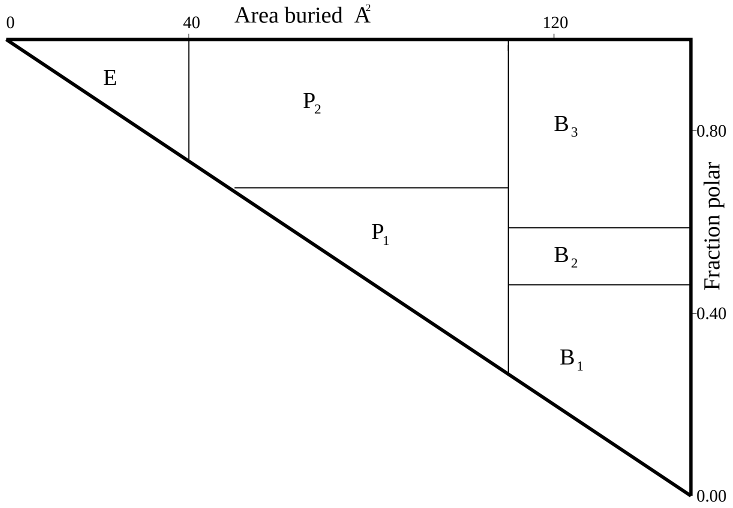


Figure 2: The six environmental classes created using the "area buried" and "fraction polar" criteria. Adopted from [12]

according to the secondary structure the residues in the respective positions adopt. The resulting environment classes discriminate buried and exposed residues, and the subdivision according to secondary structure yields a total of 18 distinct classes for the 20 amino acids. Since this leads to a $1d$ string describing the structure, it is possible to use classical alignment algorithms to find the most favorable alignment of a protein sequence to the environment string. All Eisenberg needs to do this is find a scoring matrix. To build this, a score $s = \ln\left(\frac{P_{(i:j)}}{P_i}\right)$ is defined for every amino acid on each environmental class. $P_{(i:j)}$ is the probability to find amino acid i in environment j, and P_i is the probability to find residue i in any environment. The scoring matrix (as it can be seen in Table 1), is then extracted out of a data base.

The resulting threading procedure has been successfully employed to identify sequence-structure pairs and is the basis for the threading procedure used today in many different knowledge based potentials.

class	**W**	**F**	**Y**	**L**	**I**	**V**	**M**	**A**	**G**	**P**	**C**	**T**	**S**	**Q**	**N**	**E**	**D**	**H**	**K**	**R**
$B_1\alpha$	1.00	1.32	0.18	1.27	1.17	0.66	1.26	-0.66	-2.53	-1.16	-0.73	-1.29	-2.73	-1.08	-1.93	-1.74	-1.97	-0.34	-1.82	-1.67
$B_1\beta$	1.17	0.85	0.07	1.13	1.47	1.09	0.55	-0.79	-2.02	-0.94	-0.22	-1.12	-2.91	-1.67	-1.42	-1.93	-2.56	-1.91	-2.69	-1.16
B_1	1.05	1.45	0.17	1.10	1.11	1.02	0.98	-0.91	-1.92	0.26	-1.22	-1.53	-2.81	-1.17	-2.42	-2.52	-1.76	-1.12	-2.59	-2.16
$B_2\alpha$	0.50	0.90	0.85	1.01	0.63	0.68	1.12	-0.69	-1.49	-2.21	-0.10	-1.50	-1.47	-0.23	-0.61	-0.71	-1.62	0.23	-0.78	0.06
$B_2\beta$	0.01	1.18	1.06	0.76	1.31	1.06	0.64	-1.55	-2.26	-0.49	-0.87	-2.27	-1.77	-1.22	-2.07	-1.07	-1.41	-0.77	-1.14	-0.20
B_2	1.02	1.05	1.12	0.84	0.81	0.60	0.90	-0.66	-1.66	0.19	-0.05	-0.76	-1.17	-0.76	-0.66	-1.35	-1.28	0.46	-2.34	-0.80
$B_3\alpha$	0.92	-0.03	0.58	0.15	0.04	-0.02	0.89	-0.57	-1.86	-0.68	-1.56	-0.57	-0.96	0.22	-0.06	0.08	-0.50	0.73	0.43	0.96
$B_3\beta$	0.75	0.81	1.30	0.18	0.54	0.56	-0.57	-0.93	-1.93	-0.34	-0.54	-0.44	-0.74	0.21	-0.24	-0.14	-0.86	0.82	-0.53	0.13
B_3	1.07	0.70	1.13	0.35	-0.17	-0.03	0.23	-0.96	-0.98	-0.13	-1.20	-0.53	-0.54	0.05	0.04	-0.36	-1.05	1.01	0.10	0.66
$P_1\alpha$	-1.35	-0.82	-0.59	-0.52	-0.24	0.10	-0.03	0.73	-0.49	-0.25	0.95	0.31	0.34	-0.14	-0.54	-0.17	-0.25	-0.52	-0.21	-0.28
$P_1\beta$	0.36	-0.49	0.17	-1.03	0.20	0.46	-0.27	0.64	-0.82	-0.55	1.49	0.93	0.33	-2.27	-1.32	-0.73	-1.07	-0.42	-1.21	-0.77
P_1	-1.26	-1.20	-1.31	-0.62	-0.23	-0.01	-1.19	0.46	-0.24	0.66	1.35	0.56	0.49	-0.63	-0.13	-0.61	0.38	-1.12	-0.74	-1.29
$P_2\alpha$	-1.14	-1.43	-0.79	-0.35	-0.54	-0.48	-0.45	0.06	-0.50	-0.26	-0.93	-0.05	-0.18	0.55	-0.05	0.56	0.28	0.06	0.61	0.50
$P_2\beta$	-0.79	-0.54	-0.84	-1.30	-0.33	0.13	-0.72	-0.55	-0.98	-1.29	-0.57	0.84	0.59	-0.08	-0.16	0.32	0.19	-0.87	0.59	0.10
P_2	-0.82	-0.86	-0.51	-0.70	-1.09	-0.88	-0.89	-0.15	-0.40	0.44	-0.60	0.06	0.26	0.27	0.50	0.27	0.49	0.13	0.44	0.30
$E\alpha$	-1.35	-2.20	-2.10	-1.58	-2.76	-1.10	-0.72	0.46	0.68	0.04	-0.44	-0.17	0.15	0.36	0.28	0.59	0.44	-0.19	0.13	-0.34
$E\beta$	0.64	-0.90	0.30	-1.66	-1.47	-1.74	-0.68	0.06	1.46	-0.96	-0.24	0.14	0.65	-0.19	-0.06	-0.16	-0.78	-0.83	-0.52	-0.49
E	-2.14	-1.90	-0.94	-1.19	-1.61	-0.91	-1.67	0.12	1.13	0.20	-0.46	0.12	0.32	-0.03	0.41	0.03	0.22	-0.25	-0.14	-0.32

Table 1: The scoring matrix of Eisenberg's profiling potential. Together with gap open and extension penalties, it can be used to perform alignments of sequence on structure. Eisenberg used 0.02 as open and 0.0002 as extension penalties in coiled regions, whereas in regions with defined secondary structure the open and extension penalties used are 2.0. Adopted from [12]

2.3.4 Contact Potentials

Contact potentials can be understood as subgroup of knowledge based potentials. This kind of mean energy function measures the overall energy of a system as the sum of *nearest neighbor* contact energies. They differ mainly in the definition of a contact, e.g. what atoms are chosen to represent the amino acids, and what are the criteria necessary for a contact e.g. euclidean distance only. For a discussion of the quality of different amino acid representations see [7]. The most prominent examples of contact potentials are:

Crippen's Simplified Potential

To obtain a simplified representation of hetero-polymers Ken A. Dill introduced the concept of lattice polymers [18]. When used to model proteins, each amino acids occupies one positions on the grid of the lattice. Conformations of lattice polymers are represented by *self-avoiding walks*, short SAWs. Hence this method greatly reduces the conformational space of the optimization problem. On a lattice bond lengths are, of course, always constant, furthermore potentials for lattice proteins usually neglect bond angles and dihedrals. Instead they focus on non-bonding interactions of topological neighbors.

In Crippen's potential the energy for the pair interaction is written as:

$$E(s, \mathbf{x}) = \sum_{i,j} \Psi[s(i), s(j); |i-j|; d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j)]$$

The individual interaction terms Ψ depend on the type $s(i)$ and $s(j)$ of residues, on their separation $|i-j|$ along the chain and on the euclidean distance $d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j)$ of the lattice points. Introducing $g(d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j))$ as cut off distance depending on $\mathbf{x}_i$ and $\mathbf{x}_j$, the potential function

$$\Psi[s(i), s(j); |i-j|; d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j)] = U[s(i), s(j); |i-j|] g(d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j))$$

is then normalized such that the contribution of the nearest neighbor reduces to $U[s(i), s(j); |i-j|]$.

Crippen [20] extracted a contact matrix of the form:

$$U[s(i), s(j); |i-j|] =$$
$$\begin{cases} -0.008 & \text{if } |i-j| = 3 \\ 0.004 & \text{if } |i-j| = 4 \\ 0.021 & \text{if } |i-j| = 5,6,7 \\ \begin{pmatrix} -0.012 & -0.074 & -0.054 & 0.123 \\ -0.074 & 0.123 & -0.317 & 0.156 \\ -0.054 & -0.317 & -0.263 & -0.010 \\ 0.123 & 0.156 & -0.010 & -0.004 \end{pmatrix} & \text{if } |i-j| \geq 8 \end{cases}$$

from a structural data base where the matrix entries correspond to the four amino acids classes:

$$\begin{aligned} \mathbf{1} &= \{\mathbf{G\ Y\ H\ S\ R\ N\ E}\} \\ \mathbf{2} &= \{\mathbf{A\ V}\} \\ \mathbf{3} &= \{\mathbf{L\ I\ C\ M\ F}\} \\ \mathbf{4} &= \{\mathbf{P\ W\ T\ K\ D\ Q}\} \end{aligned}$$

A further simplification of the potential can be obtained by restricting the amino acid alphabet to just two classes: **H** for hydrophobic amino acids and **P** for polar residues. For a review of **HP** based potentials see [19, 23] Crippen recently used the described potential in kinetic simulations and calculations of denaturation curves [21]. These computer experiments showed that folding kinetics largely depend on the coding scheme and that the results obtained by using the Crippen alphabet differ strongly from calculations for spin-glass encoded SAWs [27, 28].

Tropsha's Four-Point Potential

Avoiding the arbitrariness of a binned distance, A. Tropsha introduced an approach from computational geometry to knowledge based potentials [47, 66, 67]. He suggested to represent the protein structure as a set of points in

$3d$. For simplification only C_α atoms were chosen as model for the backbone. This set of points is tessellated using the *Delaunay triangulation*. The result of this geometric procedure is a partitioning of the space included by the set into irregular tetrahedrons with the points as vertices. The quadruple of amino acids represented by these points are considered to be nearest neighbors. The advantage of this method is that it is parameter free, the list of tetrahedrons is non-ambiguous.

If one counts the occurrence of all possible neighborhood combinations of the amino acids in a structural dataset, a log-likelihood function can be constructed easily. This function can then be used to test if a given sequence yields favorable contacts when threaded to a certain structure — in one word *inverse folding*.

Since the implementation of a Tropsha-based potential is the core part of this work, it will be discussed in section 2.5.4 in depth.

2.3.5 Lapedes' Neural Network `NN` Potential

Alan Lapedes *et al.* [30] developed a potential with multi-body interactions, parameterized in "local neighborhoods" for each residue. He generalized other threading approaches, and ended up in a statistical interpretation. To employ a neural net for finding a log-likelihood ratio containing higher order terms of interaction, it is necessary to find a suitable representation of the available structural information. To tackle this problem an internal coordinate system is defined, setting the C_α-atom to the center, and constructing two vectors pointing to the neighboring chain atoms: C and N. This plane has been shown to have an almost constant angle, and a third dimension is spanned by the cross product of $\overrightarrow{C_\alpha N} \times \overrightarrow{C_\alpha C}$. Further a binned sphere is constructed around the center (C_α-atom) of the coordinate system, representing a "neighborhood shell" of residues. To order this shell to spatial residues, the sphere is split into a predefined number of finite, binned sub-shells.

The chain neighbors, carrying information necessary for secondary structure, can be included as well. The M bins are filled with integers mimicking the

20 amino acids, describing the surrounding of a particular C_α atom. The neural net is trained on the PDB Select database, and parameters as number of sub-bins, bin size, or bin resolution were varied. Approaches for C_β as a core atom showed better results in threading experiments.
The usage of the internal coordinate system enables Lapedes to construct a potential not only dependent on the distance of the contacts, but also on their direction. This is achieved by dividing the spheres constructed into 8 quadrants and treating contacts differently according to what quadrants take part in the contacts.

2.3.6 Atom-Atom Potentials

While other potentials use only one point, C_α, C_β or the center of mass of an amino acid to represent a residue, atom-atom potentials use all heavy atoms of a protein, which are all atoms except Hydrogen atoms.

The reversible energy required to bring two particles close to each other at constant volume is given by the potential of mean force or Helmholtz free energy of the system. It is related to the radial distribution function $g(r)$ by:

$$w(r) = -kT \ln[g(r)]$$

and can give insights to protein folding and the role of specific interaction in native structures (e.g. H-bonds). The distribution function for arbitrary sets of atom-atom interactions occurring in proteins can either be obtained by diffraction experiments, or they are extracted from a data base of structures. The two differently obtained functions turn out to be equal, if the distance distributions are similar. The knowledge based distribution function is accessed by the determination of

$$\rho_{ab}(r) = \sum_{ab} \delta(r - r_{ij})$$

as the sum over all distinct pairs ab within the radius r in a protein library. The observed density is compared with a bulk of non interacting particles to finally obtain the distribution function:

$$g_{ab}(r) = \frac{\rho_{ab}(r)}{\rho}$$

The potentials using these distribution functions are perfectly suited for a detailed analysis of spatial distributions of atom contacts along a protein chain [48]. To make use of an atom-atom based potential, one has to know the Cartesian coordinates for *all* residues in a poly peptide chain. Therefore this approach is of no use to solve the inverse folding problem, as targeted by our group.

2.3.7 Sippl's `PROSAII` Potential

Sippl also implemented a pair potential in his software package `PROSAII` [49, 17, 50, 51]. The program was designed to determine the correctness of an experimentally derived structure under use of a quality factor *score*. The potential function used is a superposition of a pair-potential and a surface potential:

$$W(x,\psi) = \sum_{i<j} W_\gamma\left[x_i, x_j, |i-j|; \mathbf{d}^\gamma_{i,j}\right] + \sum_i V_\gamma\left[x_i; \chi(i)\right]$$

The first term W_γ stands for the pair contribution, V_γ is the surface part of the potential and both terms depend upon the backbone atom type γ (C_α or C_β). The pair-potential is calculated between amino acids x_i and x_j, located at position i and j of the sequence x. $\mathbf{d}^\gamma_{i,j}$ is the Euclidean distance of the contributing amino acids. Using a particular surface term is caused by the observation that solvent-protein interactions can be used to model amino acid energies more accurately [10, 12, 11]. The parameter χ represents a quantitative measure for the extent of surface exposure of amino acid x. The potential function as described by the parameters $W_\gamma\left[x_i, x_j, |i-j|; \mathbf{d}^\gamma_{i,j}\right]$ and $V_\gamma\left[x_i; \chi(i)\right]$ are extracted from a representative PDB-subset, applying the Boltzmann principle, and distributed with the `PROSAII`-package.

Because `PROSAII` only uses the C_β (or C_α) atoms of the backbone, and calculates the probability of finding two *residues* within a spatial distance, it is quite well suited for inverse folding studies.

2.3.8 Problems of Knowledge Based Potentials

Thomas and Dill used a lattice model with the simple **HP** differentiation to test some of the principles underlying statistical potentials [58]. Creating model proteins on a 2D lattice, they used different potentials, e.g. Sippl's, to extract the energies of contacts and compare them with the energies used to create the model structures. Some weaknesses of knowledge based potentials were discovered thus.

- The extracted energies depend on the length of the model chains.
- Most of the parameters of the knowledge based potentials seem to be redundant.
- The extracted energies depend on the amino acid composition.
- The extracted energies depend on the surface-to-volume ratio of the proteins.
- The choice of a relevant temperature for the Boltzmann distribution law is strongly dependent on the choice of proteins in the data base.
- Extracted energies can therefor only qualitatively approximate true energies.

Some of these problems have obvious reasons. The dependence of the chain length is easily explicable when using distance dependent potentials - the bigger a protein, the more long-range interactions are there. The dependence on the surface-to-volume ratio is similar - the more surface there is, the more likely it is for a **H** amino acid to be at the surface. The same is true for the amino acid composition - the more **H** a protein contains, the more likely it will be at the surface, feigning a more favorable energy for such a surface contact.

The seeming redundancy is due to the well known fact that **H** prefer the hydrophobic core of the protein. A model potential using only this information does score almost as well as much more sophisticated potentials.

Some of these problems, while they may be grave, do not really affect the main applications of knowledge based potentials. If the threading procedure and the z-score (Chapter 2.4) is used, the Boltzmann temperature only scales the energies and does not affect their relative order. Since the extracted energies are only compared to other extracted energies, the lack of physical accuracy can also be neglected.

For other problems, like the dependence on amino acid composition, chain length and surface-to-volume ratio, it must be attempted to keep the errors small by using a data base as diverse as possible.

2.4 Energy and Z-score

Statistical analysis of the Delaunay tessellation of a protein yields the q factors for the occurring quadruples as the likelihood of finding this particular contact within the structure. Based on equation 4, it is possible to define the energy of a sequence S on a fold ψ as the sum over the log-likelihoods of all contacts that occur in ψ:

$$W(S, \psi) = \sum_{contacts} q_{contact} \tag{2}$$

where $q_{contact}$ is the statistic likelihood of a quadruple.

However since the determination of the ground state for each sequence would require to solve the folding problem, it is not possible to normalize the energy function. But defining a quantity called z-score as an energy separation between the native fold and the average of an ensemble of mis-folds in the units of standard deviation of the ensemble, can be used for constructing an energy scale by which conformations between different sequences can be compared. Following Sippl [17, 49, 50, 51] we define

$$z(S,\psi) = \frac{W(S,\psi) - \overline{W}(S)}{\sigma_{W(S)}} \tag{3}$$

where $\overline{W}(S)$ is the average energy of sequence S in all conformations of a data base and $\sigma_W(S)$ denotes the standard deviation.

The data base has to be a source of alternative conformations for the sequence S with length N. If the data base size x of possible structures is set to a fixed number, the number of possible decoys is a function of the sequence length l. So for the limit $l \rightarrow N$ the data base becomes insignificant. This problem has been circumvented by the construction of a “poly-protein” by linking all structures that are initially constructed for the measurement of the log-likelihood.

The sequence of the protein to be tested is slid along this aggregate of proteins from the N- to the C-terminus of the structural library amino acid by amino acid. For each aligned structure a z-score is calculated and counted as “mis-fold” to the ensemble, therefore it does not make too much sense to use a member of the dataset for testing the threading capabilities of the potential via the z-score . If $n \sim 40.000$ is the length of the poly-protein, $n - l$ miss-folds can be constructed. Since $n \gg l$ this number of sequence-structure pairs is of the same order of magnitude as the poly-protein length. This computational brute force attack is sufficient if it is not necessary to have gaps within the sequence-structure alignment. Otherwise more sophisticated techniques must be used.

An experimental test of the z-score using thermodynamic data could demonstrate the definite significance of the scale [65]. A z-score range from 15-30 for small native proteins could be observed. The magnitude of these scores shows the need to improve existing potentials. The scores derived from existing potentials are in the range of 5-20, so there is still room for improvement.

2.5 Delaunay Tessellation

As mentioned above, A. Tropsha introduced a computational geometry approach, Delaunay Tessellation, to knowledge based potentials.
The common meaning of *"tessellation"* is to arrange squares in a mosaic pattern. The term derives from the Greek word *"tesseres"* which means four. Generally *tessellating* can be understood as arranging regular polygons congruently (all angles and sides are equal) in a plane with edges attached to each other. Only three regular polygons tessellate in the Euclidean plane: triangles, squares and hexagons (Figure 3). By extension, higher dimensional spaces can also be tessellated.

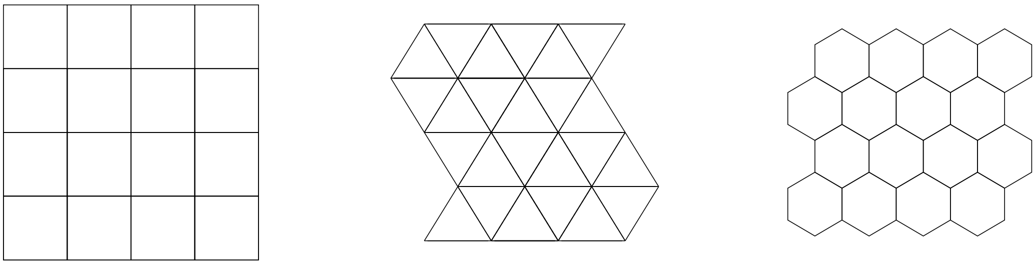


Figure 3: Tessellations in two dimensions. Figure adopted from [61]

The Delaunay triangulation *tessellates* a set of points in R^3 in the sense of filling space with tetrahedrons. The Delaunay triangulation is computed via its dual, the Voronoi diagram.

2.5.1 The Voronoi Diagram

Given a set S of n distinct points in $\mathbb{R}^d$, a Voronoi diagram is the partition of $\mathbb{R}^d$ into n polyhedral regions $\mathrm{vo}(p), (p \in S)$. Each region $\mathrm{vo}(p)$, called the Voronoi cell of p, is defined as the set of points in $\mathbb{R}^d$ which are closer to p than to any other points in S, or more precisely,

$$\mathrm{vo}(p) = \{x \in \mathbb{R}^d | \mathrm{dist}(x,p) \leq \mathrm{dist}(x,q) \forall q \in (S-p)\}$$

where dist is the Euclidean distance function. The set of all Voronoi polyhedrons forms a cell complex. The vertices of this complex are called the

Voronoi vertices, and the extreme rays (i.e. unbounded edges) are the *Voronoi rays.*

For each point $v \in \mathbb{R}^d$, the *nearest neighbor* set $\mathrm{nb}(S, v)$ of v in S is the set of points $p \in S - v$ which are closest to v in Euclidean distance. In order to compute the Voronoi diagram, the following construction is very important. For each point p in S, consider the hyperplane tangent to the paraboloid in $\mathbb{R}^{d+1}$: $x_{d+1} = x_1^2 + \cdots + x_d^2$. This hyperplane is represented by $h(p)$:

$$\sum_{j=1}^{d} p_j^2 - \sum_{j=1}^{d} 2p_j x_j + x_{d+1} = 0$$

By replacing the equality with inequality $\geq$ above for each point p, we obtain the system of n inequalities, which we denote by $b - Ax \geq 0$. The polyhedron P in $\mathbb{R}^{d+1}$ of all solutions x to the system of inequalities is a lifting of the Voronoi diagram to one higher dimensional space. In other words, by projecting the polyhedron P onto the original $\mathbb{R}^d$ space, we obtain the Voronoi diagram in the sense that the projection of each facet of P associated with is exactly the Voronoi cell $\mathrm{vo}(p)$. The vertices and the extreme rays of P project exactly to the Voronoi vertices and the rays, respectively.

2.5.2 Delaunay Triangulation

Let S be a set of n points in $\mathbb{R}^d$. The convex hull $\mathrm{conv}(nb(S, v))$ of the nearest neighbor set of a Voronoi vertex v is called the Delaunay cell of v. The Delaunay complex (or triangulation) of S is a partition of the convex hull $\mathrm{conv}(S)$ into the Delaunay cells of Voronoi vertices.

The Delaunay complex is not in general a triangulation but becomes a triangulation when the input points are non-degenerate, i.e. no $d + 2$ points are co-spherical or equivalently there is no point whose nearest neighbor set has more than $d + 1$ elements. The Delaunay complex is dual to the Voronoi diagram in the sense that there is a natural bijection between the two complexes which reverses the face inclusions.

There is a direct way to represent the Delaunay complex, just like the Voronoi diagram. In fact, it uses the same paraboloid in $\mathbb{R}^{d+1} : x_{d+1} = x_1^2 + \cdots + x_d^2$. Let $f(x) = x_1^2 + \cdots + x_d^2$, and let $\tilde{p} = (p; f(x)) \in \mathbb{R}^{d+1}$ for $p \in S$. Then the so-called lower hull of the lifted points represents the Delaunay complex. More precisely, let

$$P = \text{conv}(\tilde{S}) + \text{noneg}(e^{d+1})$$

where e^{d+1} is the unit vector in $\mathbb{R}^{d+1}$ whose last component is 1. Thus P is the unbounded convex polyhedron consisting of conv$(\tilde{S})$ and any nonnegative shifts by the "upper" direction r. The nontrivial claim is that the boundary complex of P projects to the Delaunay complex: any facet of P which is not parallel to the vertical direction r is a Delaunay cell once its last coordinate is ignored, and any Delaunay cell is represented this way.

Considering a set of point in $\mathbb{R}^3$ the Delaunay triangulation describes an algorithm to decompose the convex hull of these points into tetrahedrons.

2.5.3 The `qhull` Algorithm

As previously described, the first step in generating the tessellation built from the irregular tetrahedron is finding the convex hull, which is the smallest convex set of points containing the entire set. The hull is represented by a set of facets and a set of adjacency lists giving the neighbors and vertices for each facet. In $\mathbb{R}^3$ facets are triangles and ridges are edges. The Delaunay triangulation in $\mathbb{R}^d$ is calculated from a convex hull in $\mathbb{R}^{d+1}$ by lifting the points to a paraboloid by adding the sum of the squares of the coordinates and computing their convex hull, the set of ridges of the lower convex hull is the Delaunay triangulation of the original set.

The `qhull` algorithm [13] is a variation of the randomized incremental algorithm, employing a constructed additional point at the hull to decide which facet belongs to it. The point is outside the facet if it is above the set and in the `qhull` variation of the original version, the point is not created randomly, but at the furthest distance from the outside set. This method is used in the program `qhull` which is publicly available via the Internet[1]. It has been

---

[1] `http://www.geom.uiuc.edu/software/download/qhull.html`

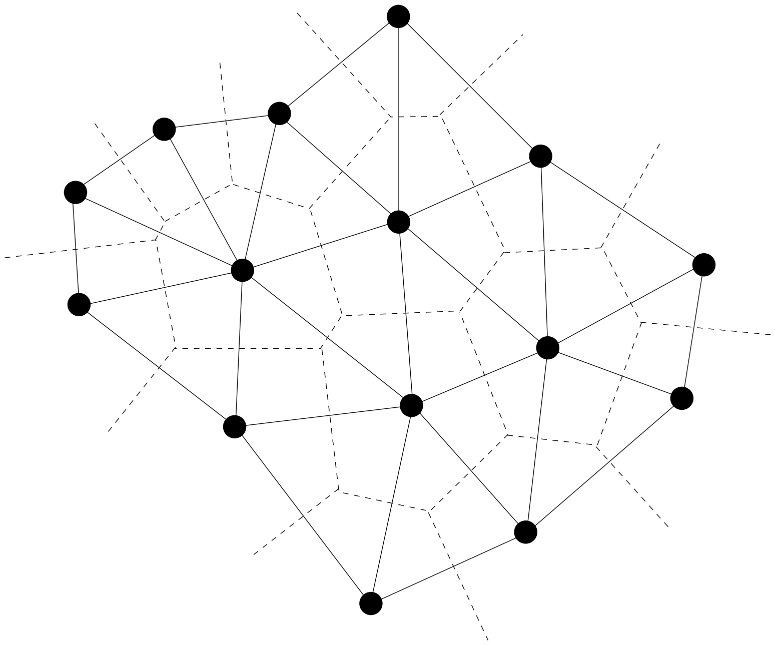


Figure 4: Voronoi diagram of a set of points in $2d$: The Delaunay triangulation can easily be computed via its dual, the nearest neighbors of each Voronoi vertex are connected in the Delaunay diagram. Voronoi cells are shown with dashed lines. Figure adopted from [61]

shown empirically that this algorithm is especially efficient and well suited for a $3d$ set of points[13].

This algorithm of triangulation can be applied to any set of points in space, always objectively describing neighborhood. Representing amino acids of a poly-peptide chain by an atom (e.g. C_α or C_β) leads to a regular set of points in $3d$ space. This set can be tessellated applying the rules described above. The Voronoi polyhedron is now the region around an atom, each side describes a contact to a neighbor. The underlying Delaunay simplices are irregular tetrahedrons with an amino acid positioned at each corner. This diagram can be employed to describe contacts of amino acids objectively in $3d$ space. Figure 5 shows the Delaunay tessellation of a protein.

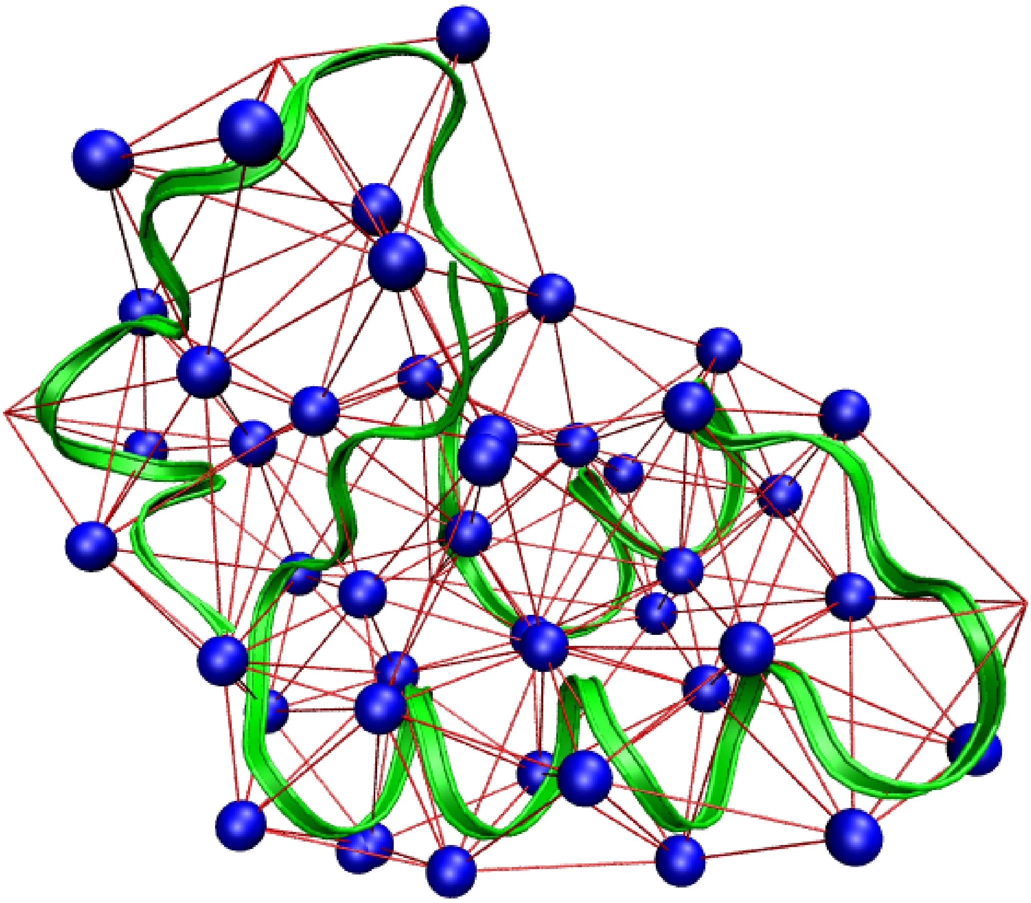


Figure 5: Delaunay tessellation of crambin (pdb-code 1ab1), the tessellation was computed using a water shell, for clearance the tetrahedrons of the water shell have been omitted. Green is the ribbon representation of the backbone, blue are the C_β atoms, red the Delaunay tetrahedrons.

2.5.4 Empirical Protein Potentials from Delaunay Tessellation

2.5.5 Four Body Contact Potentials

Based on the fact that four *neighboring* points in space form an irregular tetrahedron, applying the Delaunay tessellation to a set of points in $3d$ results in a contact potential. The likelihood of finding a distinct set of labeled points in this set can be expressed as:

$$q_{ijkl} = \log \frac{f_{ijkl}}{p_{ijkl}} \tag{4}$$

where i, j, k, l are four amino acids, f_{ijkl} is the *observed* normalized frequency of occurrence of a given quadruple, and p_{ijkl} is the *a priori expected* frequency of occurrence of a given quadruple. So q_{ijkl} is a measurement of likelihood for finding four distinct amino acids in a simplex, namely a log-likelihood. The observed frequency f_{ijkl} is calculated by dividing the total number of occurrences of each quadruple by the number of all observed quadruples.

$$p_{ijkl} = \beta a_i a_j a_k a_l \tag{5}$$

where a_i, a_j, a_k, and a_l denote the individually observed frequency of occurrence of each amino acid residue. That is the total number of occurrence of a distinct amino acid type divided by the total number of residues in the dataset. β is the combination factor, accounting for the fact that replicated residue types are underestimated due to permutability. β is defined as (refer to appendix (B) for explicit values of β):

$$\beta = \frac{4!}{\prod_i^k t_i!} \tag{6}$$

with k being the number of distinct residue types in a quadruple and t_i is the number of amino acids of type i.

Applying this procedure to a predefined set of experimentally derived protein structures leads to a potential of mean force. The calibration dataset has to be selected with care, since this selection determines the discriminative power of the force field. Parameters like the protein type (e.g. globular, membrane, soluble etc), the type of backbone atom used for tessellation and any kind of selection of tetrahedrons (i.e. filtering) have to be kept constant for the parameter set.

Besides using a simple four point contact potential

$$E_{i,j,k,l} = \frac{f^{exp}_{i,j,k,l}}{f^{obs}_{i,j,k,l}},$$

where every amino acid is represented by its C_α atom Carter *et al.* [16] have shown recently that this type of potential can be used for correlating stability and hydrophobic core mutations. They used the center of mass of the amino acids instead of the C_α and the tetrahedrons of the hydrophobic core exclusively for modeling the energy differences caused by mutations in the hydrophobic core. The correlation to experimental results they achieved was better than that of distance-derived statistical potentials. That seems to back the physical validity of this type of four point contact potentials. Weberndorfer *et al.* improved the performance of a four point potential by dividing it into a core and a surface potential and using the C_β for amino acid representation [62]. They also tried to improve the statistics of this approach by using smaller amino acid alphabets.

3 The `SaDSaT` Potential

Compared to Sippl's `PROSAII` potential, the 4 point potentials introduced by Tropsha and used by Weberndorfer and Carter neglect sequence separation and euclidean distances, which were shown to have a significant impact on the quality of Sippl's potential. Therefore, we try to introduce these two descriptors as well as a different way to treat the surface interactions into a tessellation potential, the `SaDSaT` (**S**equence **a**nd **D**istance **S**eparation **a**pplied to **T**essellation) potential.

For `SaDSaT` we use C_β atoms only to represent the residues of the protein. While keeping the number of points to be tessellated small, usage of C_β instead of C_α gives at least some information about the position of the side chain represented by the C_β. Of course for Glycine, lacking a C_β atom, it has to be designed.

3.1 Overview of Innovations

3.1.1 Distance Classes

In principle, introducing sequence and euclidean distance classes is straightforward, but simply adding these classes to the tetrahedrons obtained by tessellation is not feasible. There are 8855 different combinations of 4 amino acids and hence different possible tetrahedrons for the tessellation. But even if we were to distinguish only 4 classes at the edges, the number of possible tetrahedrons rises to 6455295. Obviously, there is not enough data available to do reliable statistics on that. In our database, the total number of tetrahedrons is about 1.5 million. So we cannot use this more detailed approach on the tetrahedrons.

On the other hand, there is sufficient data to distinguish an even bigger number of classes at pair level, so, to overcome those difficulties, we try to split the potential of the tetrahedrons.

Joint probabilities can be expressed as conditional probabilities as follows:

$$\log p(A \wedge B \wedge C) = \log p(A \wedge C) + \log p(B|A \wedge C)$$

That can be the starting point for an approximation, if B is considered to be independent of C. We then assume that:

$$\log p(B|A \wedge C) \approx \log p(B|A)$$

and

$$\log p(A \wedge B \wedge C) \approx \log p(A \wedge C) + \log p(B|A)$$

While it may be infeasible to sample A, B and C together, we now have only A and C as well as $B|A$ to consider. To apply this to our tessellation, we first decompose our 4 point potential into pair, triangle and tetrahedron terms:

$$\begin{aligned}\sum_{\text{tetrahedron}} q_{tetrahedrons} &= \sum_{\text{AAs}} q^*_{AA} + \\ &\sum_{\text{pairs}} q^*_{pair} + \\ &\sum_{\text{triangles}} q^*_{triangle} + \\ &\sum_{\text{tetrahedrons}} q^*_{tetrahedron}\end{aligned}$$

where the q^* are dependent probabilities, i.e. $q^*_{tetrahedron}$ is now dependent on the observed frequencies (f_{obs}) of the triangles, the same is true for triangles and pairs and pairs and amino acids.

After the decomposition, we can now begin to deal with the individual terms. In each term, we will define an energy contribution as

$$\Delta E = \log(f_{obs}/f_{exp})$$

getting a positive energy for events that happen more often than expected. The sign is of course exactly wrong physically speaking, but more convenient for computation.

3.1.2 Surface Terms

The surface of a globular protein is very different to its core. Generally, the amino acids in the core are the hydrophobic ones, whereas hydrophilic amino acids tend to be at the surface. In contrast to Weberndorfer's approach [62], which basically distinguishes between surface and core tetrahedrons, we decided to use explicit water molecules as shown in [68] to simulate surface interactions. The water was incorporated as 21st "amino acid" into the potential, although we will later see that it was not treated exactly like the other 20.

While finding the parts of tetrahedrons which are on the surface is a yes/no differentiation, using explicit water molecules can give us additional information concerning the degree of surface exposure. However, it is important to remember that because only C_β atoms are used for generating the virtual solvent molecules respectively the surface tetrahedrons, only a crude approximation of the surface can be achieved. It is crucial for the performance of a potential to carefully balance the contribution of the surface term, no matter how it is computed, and the hydrophobic interactions in the core of the protein.

3.2 Pairs

The pair-term, with only 210 different pairs possible, is the term where different sequence or euclidean classes can be applied while retaining sufficiently reliable statistics. Following Sippl, we divide the continuous euclidean distances of the C_βs into distinguished classes, as well as assigning different classes for the sequential separation of the contacts. Note that the difference to Sippl's `PROSAII` potential is simply the definition of a contact, here being amino acids participating in the same tetrahedron.

The expected frequency of an amino acid pair a, b, in the sequence distance class x and the euclidean distance class y is:

$$f^{exp}_{a,b,x,y} = \beta_{a,b} p_a p_b \frac{\sum_{\mathrm{a},b} N_{a,b}}{\sum_{\mathrm{a},b} N_{a,b,x,y}} \tag{7}$$

with $N_{a,b}$ the number of pairs a, b in the database, $N_{a,b,x,y}$ the number of pair a, b in the sequence class x and the euclidean class y in the database, p_a and p_b are the frequencies of the amino acids in the data base and the binomial coefficient β (Appendix B) necessary because there is no way to distinguish between a pair a, b and a pair b, a.
While Sippl's potential uses 20 euclidean distance classes with a size of 2 Å [32], due to the nature of the tessellation, where the maximum distance between C_βs is about 10 Å, the number of euclidean distance classes sensibly used here will be a lot smaller, namely two to three distance classes of various length (Chapter 4.1.3).

3.3 Triangles

In order to compute the expected frequencies of a triangle dependent on the frequencies of the pairs, the natural choice are expected frequencies which conform to the Principle of Maximum Entropy.

3.3.1 Principle of Maximum Entropy

The formulation of a pair-dependent expected frequency of triangles leads to an optimization problem. We use the so called Maximum Entropy Principle as a means to assign numerical values to probabilities we have certain information about. It states that, given a set of n mutually exclusive events, the probability values $p(n)$ have to be chosen such that the (Shannon) entropy of the distribution $p = (p_1, ..., p_n)$, i.e. the expression

$$H(p) = -\sum_i p_i \ln p_i \tag{8}$$

attains its maximum value under the condition that p agrees with the information given. The boundary conditions are usually integrated using Lagrange multipliers λ. So for the boundary condition $\sum_i p_i = 1$

$$H(p) = -\sum_i p_i \ln p_i + \lambda_i \Big(\sum_i (p_i) - 1 \Big) \tag{9}$$

is used.

3.3.2 Application

With the maximum entropy approach we use equation (9) to maximize the entropy of the expected frequencies of the triangles, $P_{a,b,c}$, with the boundary conditions that the triangle frequencies are dependent of the pair frequencies, i.e. that

$$\begin{aligned} \sum_c P_{a,b,c} &= P_{a,b} \\ \sum_a P_{a,b,c} &= P_{b,c} \\ \sum_b P_{a,b,c} &= P_{a,c} \end{aligned}$$

where $P_{a,b}$, $P_{a,c}$ and $P_{b,c}$ are the frequencies of the respective pairs in the data base. This leads to

$$\begin{aligned} H = & \sum_{a,b,c} P_{a,b,c} \ln P_{a,b,c} \\ + & \lambda_{a,b} \Big(\sum_c (P_{a,b,c}) - P_{a,b} \Big) \\ + & \lambda_{b,c} \Big(\sum_a (P_{a,b,c}) - P_{b,c} \Big) \\ + & \lambda_{a,c} \Big(\sum_b (P_{a,b,c}) - P_{a,c} \Big) \rightarrow \max \end{aligned} \tag{10}$$

or, for easier differentiation:

$$\begin{aligned} H = & \sum_{a,b,c} P_{a,b,c}\big(\ln P_{a,b,c} - 1\big) \\ + & \lambda_{a,b}\Big(\sum_{c}(P_{a,b,c}) - P_{a,b}\Big) \\ + & \lambda_{b,c}\Big(\sum_{a}(P_{a,b,c}) - P_{b,c}\Big) \\ + & \lambda_{a,c}\Big(\sum_{b}(P_{a,b,c}) - P_{a,c}\Big) \rightarrow \max \end{aligned} \tag{11}$$

so:

$$\frac{\partial H}{\partial P_{a,b,c}} = \ln P_{a,b,c} + \lambda_{a,b} + \lambda_{b,c} + \lambda_{a,c} = 0 \tag{12}$$

and

$$P_{a,b,c} = e^{\lambda_{a,b}} e^{\lambda_{b,c}} e^{\lambda_{a,c}} \tag{13}$$

We did not succeed in solving this problem analytically, but because of

$$\frac{\partial^2 H}{\partial P^2_{a,b,c}} = \frac{1}{P_{a,b,c}}$$

we know this equation to have only one maximum.
So we used a maximizer, `donlp2`[2] [54, 55], which uses a sequential quadratic programming method, to numerically maximize

$$H = \sum_{a,b,c} P_{a,b,c}\big(\ln P_{a,b,c} - 1\big) \tag{14}$$

using 210 boundary conditions of the type

$$P_{a,b} = e^{\lambda_{a,b}} + \sum_{c} \left(e^{\lambda_{a,c}} e^{\lambda_{b,c}}\right) \tag{15}$$

`donlp2` switches between two different approaches for convergence to the solution; A fast if not that reliable way if certain conditions hold, and a slow but reliable way if they don't. This leads to a fast, reliable convergence of

---

[2] `ftp://ftp.mathematik.tu-darmstadt.de/pub/department/software/opti/`

the solutions.
Since it is not possible to analytically solve this type of problems, testing the results is not possible directly. However, we tested the results of a "regular" subtype of problems, i.e. problems where $P_{a,b} = \beta P_a P_b$ and found `donlp2` to be able to solve these problems correctly. Another test we performed is using different starting values for the computations. The results were almost equal, differing only at the ‰ level, only the time needed for computation changed. While these tests indicate that the results we obtained are plausible, directly testing them would be preferable.
To achieve this, another approach to solve this type of maximum entropy problem, called iterative proportional fitting (e.g. [25]), could be used. This procedure is a calibration technique for estimating cell frequencies of a matrix (in our case the $p_{triangles}$) using known marginals (in our case the p_{pairs}) under the condition that the marginals are reproduced. It is implemented as a three step iterative algorithm starting with equal distribution. Convergence with respect to our kind of problem has been shown ([22, 38]). The iterative proportional fitting procedure can be visualized as trying to find the intersecting line of two planes. Starting somewhere above that line, alternating projections onto the two planes will yield a zig-zag curve converging to the intersection. In this picture, the planes stand for sets of distributions having one of the required marginals.

3.4 Tetrahedrons

In principle, the expected frequency of tetrahedrons can be computed similar to that of the triangle part, but instead of 400 or, considering the symmetry of the problem, 210 different exponents we would have to compute 8000, which, alas, is not possible with `donlp2`.

3.4.1 Computation using Triangles

To come up with a triangle dependent expected frequency of a tetrahedron, we begin with solving the regular case, i.e. the case where the pair frequencies $P_{a,b}$, the triangle frequencies $P_{a,b,c}$ and the tetrahedron frequencies $P_{a,b,c,d}$ are all regularly computed out of the amino acid frequencies p_a, using β, the binomial coefficient first introduced in chapter 2.5.5:

$$\begin{aligned} P_{a,b} &= \beta_{a,b} p_a p_b && (16) \\ P_{a,b,c} &= \beta_{a,b,c} p_a p_b p_c && (17) \\ P_{a,b,c,d} &= \beta_{a,b,c,d} p_a p_b p_c p_d && (18) \end{aligned}$$

To get an expression for the tetrahedron frequencies dependent on the triangle frequencies, we write

$$P_{a,b,c,d} = \beta_{a,b,c,d} P_{a,b,c} P_{a,c,d} P_{a,b,d} P_{b,c,d} \cdot T_{a,b,c,d} \tag{19}$$

with $T_{a,b,c,d}$ a factor we now try to compute using equation (17).
We obtain

$$P_{a,b,c,d} = \beta_{a,b,c,d} \beta_{a,b,c} p_a p_b p_c \beta_{a,b,d} p_a p_b p_d \beta_{a,c,d} p_a p_c p_d \beta_{b,c,d} p_b p_c p_d T_{a,b,c,d} \tag{20}$$

and

$$P_{a,b,c,d} = \beta_{a,b,c,d} \beta_{a,b,c} \beta_{a,b,d} \beta_{a,c,d} \beta_{b,c,d} T_{a,b,c,d} p_a^3 p_b^3 p_c^3 p_d^3 \tag{21}$$

Since $P_{a,b,c,d} = \beta_{a,b,c,d} p_a p_b p_c p_d$, we can compute $T_{a,b,c,d}$:

$$T_{a,b,c,d} = \frac{1}{\beta_{a,b,c} \beta_{a,b,d} \beta_{a,c,d} \beta_{b,c,d} p_a^2 p_b^2 p_c^2 p_d^2} \tag{22}$$

or

$$T_{a,b,c,d} = \frac{1}{\sqrt[3]{\beta_{a,b,c} \beta_{a,b,d} \beta_{a,c,d} \beta_{b,c,d} (P_{a,b,c} P_{a,b,d} P_{a,c,d} P_{b,c,d})^2}} \tag{23}$$

For the expected frequency of the tetrahedron in the regular case we then get:

$$P_{a,b,c,d} = \frac{\beta_{a,b,c,d}}{\sqrt[3]{\beta_{a,b,c} \beta_{a,b,d} \beta_{a,c,d} \beta_{b,c,d}}} \sqrt[3]{P_{a,b,c} P_{a,b,d} P_{a,c,d} P_{b,c,d}} \tag{24}$$

with the binomial coefficients β listed in appendix(B).

However, very small problems of this type can be solved numerically, and the results show that the regular case can not be applied here. Because $P_{a,b,c}$ is not linearly dependent on p_a, p_b and p_c, we try to adapt the expected frequencies by trying to find a correction similar of form to e.g. Van der Waals gas laws in thermodynamics [59].

Trial and error lead to a correction of the form $P'_{a,b,c,d} = P_{a,b,c,d} f_{corr}$:

$$\begin{aligned} f_{corr} &= \frac{\beta_{a,b,c} \sum_{b,c} P_{a,b,c} \sum_{a,c} P_{a,b,c} \sum_{a,b} P_{a,b,c}}{P_{a,b,c}} \\ &+ \frac{\beta_{a,b,d} \sum_{b,d} P_{a,b,d} \sum_{a,d} P_{a,b,d} \sum_{a,b} P_{a,b,d}}{P_{a,b,d}} \\ &+ \frac{\beta_{a,c,d} \sum_{c,d} P_{a,c,d} \sum_{a,c} P_{a,c,d} \sum_{a,d} P_{a,c,d}}{P_{b,c,d}} \\ &+ \frac{\beta_{b,c,d} \sum_{c,d} P_{b,c,d} \sum_{b,c} P_{b,c,d} \sum_{b,d} P_{b,c,d}}{P_{b,c,d}} \end{aligned} \tag{25}$$

Where f_{corr} is to be understood as a measure of how much the triangle terms deviate from the regular case. We were not able to explain why this correction factor is better than any other, or even why it should be applied in this particular way in the first place.

Because this condition was extremely unsatisfying, we tried to work out a way around this problem:

3.4.2 Tetrahedrons dependent on Pairs

Since the usage of triangles is not compulsory, we decided to omit the triangle term and use the following approach:

$$\begin{aligned} \sum_{\text{tetrahedron}} q_{\text{tetrahedrons}} &= \sum_{\text{AAs}} q^*_{AA} \\ &+ \sum_{\text{pairs}} q^*_{pair} \\ &+ \sum_{\text{tetrahedrons}} q^*_{tetrahedron} \end{aligned} \tag{26}$$

where the $q^*_{tetrahedron}$ are now dependent on the pair frequencies. As done with the triangles above, we compute the expected frequencies using `donlp2`.

$$\begin{aligned}
H &= \sum_{\mathrm{a},b,c,d} P_{a,b,c,d}(\ln P_{a,b,c,d} - 1) \\
&+ \lambda_{a,b}\Big(\sum_{c,d}(P_{a,b,c,d}) - P_{a,b}\Big) \\
&+ \lambda_{a,c}\Big(\sum_{b,d}(P_{a,b,c,d}) - P_{a,c}\Big) \\
&+ \lambda_{a,d}\Big(\sum_{b,c}(P_{a,b,c,d}) - P_{a,d}\Big) \\
&+ \lambda_{b,c}\Big(\sum_{b,d}(P_{a,b,c,d}) - P_{b,c}\Big) \\
&+ \lambda_{b,d}\Big(\sum_{a,c}(P_{a,b,c,d}) - P_{b,d}\Big) \\
&+ \lambda_{c,d}\Big(\sum_{a,b}(P_{a,b,c,d}) - P_{c,d}\Big) \rightarrow \max
\end{aligned} \tag{27}$$

$$\frac{\partial H}{\partial P_{a,b,c,d}} = \ln P_{a,b,c,d} + \lambda_{a,b} + \lambda_{a,c} + \lambda_{a,d} + \lambda_{b,c} + \lambda_{b,d} + \lambda_{c,d} = 0 \tag{28}$$

and therefore:

$$P_{a,b,c,d} = e^{\lambda_{a,b}} e^{\lambda_{a,c}} e^{\lambda_{a,d}} e^{\lambda_{b,c}} e^{\lambda_{b,d}} e^{\lambda_{c,d}} \tag{29}$$

Again, the solution is unique since H is concave

$$\frac{\partial^2 H}{\partial P^2_{a,b,c,d}} > 0$$

and

$$\frac{\partial^2 H}{\partial P_{a,b,c,d} \partial P_{a',b',c',d'}} = 0$$

if

$$(a, b, c, d) \neq (a', b', c', d')$$

The numerical solution of the system of equations is feasible using `donlp2` because while there is a great number of $P_{a,b,c,d}$, there are only 210 e^λs. The results computed by `donlp2` were tested again by using the regular case and different starting values. Again, tests indicated that the results produced by `donlp2` are plausible.

3.5 Surface

3.5.1 Water Shell

Weberndorfer *et al.* [62] used a simple approach to determine whether an amino acid is part of the surface of a protein or not.
The face of a tetrahedron is either shared with another tetrahedron, or it is part of the surface. Thus, every amino acid being part of such a face of a tetrahedron is considered to belong to the surface of the protein. Instead of this yes/no type of decision, we tried to work out a way to not only consider whether an amino acid is part of the surface, but also the degree of surface exposure. So, following Zimmer [68], we introduced an artificial solvent to our computations.

To simulate the surface interactions between the protein and its solvent, we use a "shell" of virtual water molecules. This shell is generated after reading the coordinates of the protein backbone. The generated water molecules are then tessellated along with the amino acids.
When doing this, it is important to find a compromise between the resolution of the water shell and the necessary computational effort. Too many water molecules raise the time of computation and the memory capacities needed significantly, while a too coarse grained water shell will give a poor resolution of the surface. A big shortcoming of using only one atom to represent an amino acid is also that there is no way to tell where the side chains of the amino acids are, making it impossible or at least very improbable to correctly assign all the surface amino acids and find cavities within the correctly shaped protein. As an artificially introduced cavity would cause a big error in the potential, the probability to miss some amino acids which are part of the

surface has to be accepted.

In order to generate a water shell, we first generate a box filled with water on a cubic grid. The box is made by taking the maximum extension of the protein, expanding it by $2(d_{min} + 2g_{hoh})$, the minimum distance between water molecules and C_β and the distance and 2 times the grid distance. We get a box of water with dimensions:

$$\begin{aligned}
x^{box}_{\mathrm{min}} &= x^{protein}_{\mathrm{min}} - (d_{\mathrm{min}} + 2g_{hoh}) \\
x^{box}_{\mathrm{max}} &= x^{protein}_{\mathrm{max}} + (d_{min} + 2g_{hoh}) \\
y^{box}_{\mathrm{min}} &= y^{protein}_{\mathrm{min}} - (d_{min} + 2g_{hoh}) \qquad (30) \\
&\vdots
\end{aligned}$$

The second step is removing all the water molecules which get closer than the cutoff distance d_{min} to the C_β atoms of the protein, generating a roughly "protein shaped" hole in the water grid. Then all the water molecules too far away from the protein to participate in tetrahedrons with amino acids are deleted. To achieve this, all water molecules where all neighboring grid positions are still occupied by water are deleted. That will only be the case

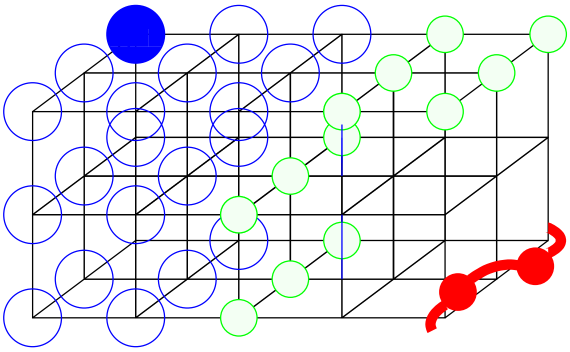


Figure 6: Illustration of the water eliminating procedure. Red is the Protein backbone with the C_βs, the surrounding water molecules have been expelled. Starting at the filled blue sphere, all blue water molecules are expelled according to the rule stated above, leaving only the green ones.

if these water molecules have not been removed due to the distance-to-the-protein criterion (Figure 6).
Finally, because the tessellation of a cubic grid is not possible as you would get cubes instead of tetrahedrons, (Figure 7), the coordinates of the water molecules are slightly perturbed, randomly choosing the direction as well as the length (up to 2Å) of the translation.
The result of this procedure is a shell of waters with a depth of approximately 2, as can be seen in Figure 8.

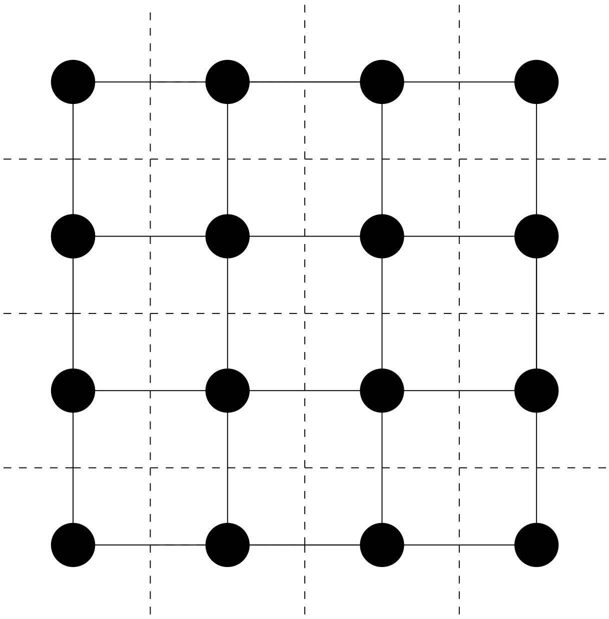


Figure 7: Attempt to Delaunay tessellate a square grid. As can be seen, the results are not triangles but squares. By extension, this also applies in 3 Dimensions.

The number of water molecules added depends on the length and surface of the protein to be tessellated. For globular proteins, the ratio of the surface to the volume, and hence to the number of amino acids, is expected to be $\propto N_{aa}^{\frac{2}{3}}$. As can be seen in figure 9, the proteins used in our data base follow this expectation.

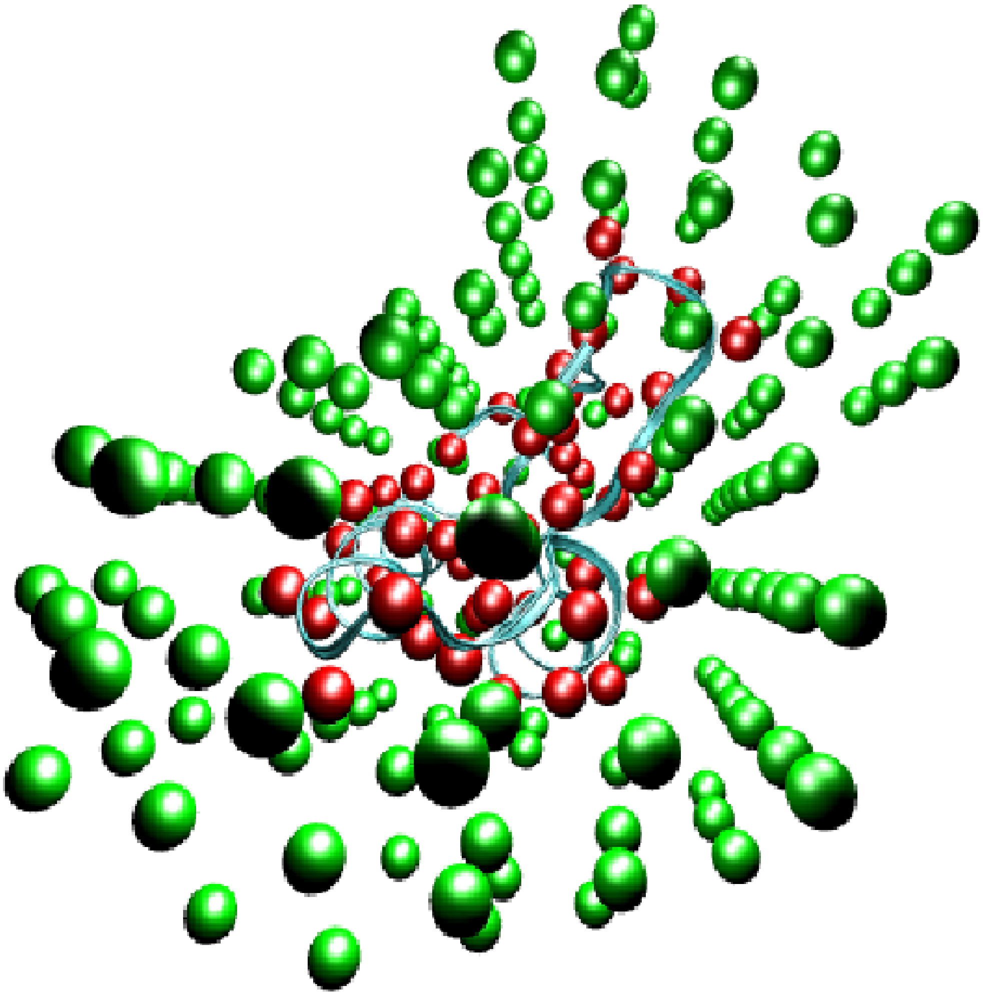


Figure 8: The water-shell of crambin (PDB-code 1ab1). Red are the C_βs of the protein, green the molecules of the water shell

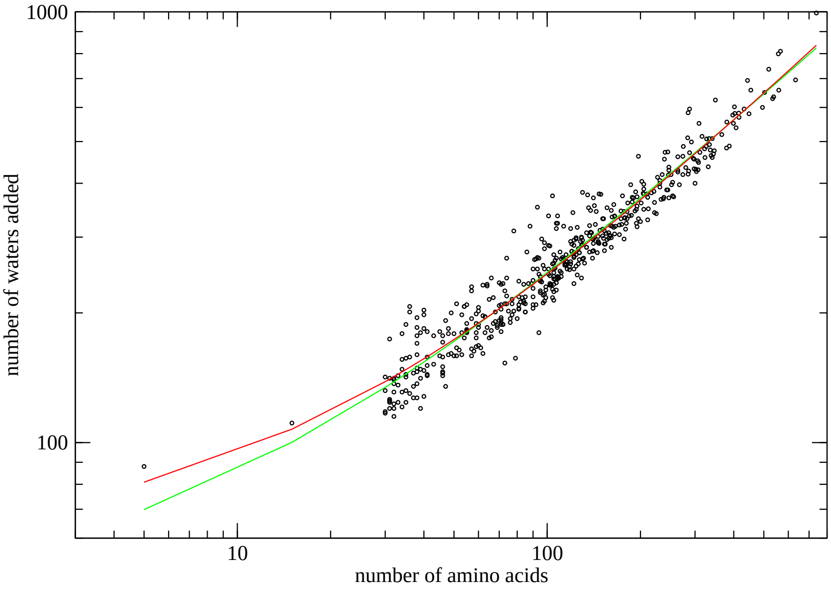


Figure 9: Number of water molecules added to proteins of our data base. Red: fitted regression $\propto N_{aa}^{0.7}$ (low number of amino acids is weighted higher) green: regression $\propto N_{aa}^{\frac{2}{3}}$

When using the threading procedure and thus producing non-native, non-globular shapes, the number of water molecules created can exceed five times the number of amino acids.

3.5.2 Expected Frequencies of Surface Contacts

All the expected and observed frequencies used above are frequencies computed or extracted from the data base without counting amino acid - water contacts. The contribution of the surface to the energy is computed in a different way.
Firstly, because water is not part of the sequence it does not make sense to use sequence distance classes when computing the contributions of water - amino acid pairs. Neither is a classification based on euclidean distance applicable since the location of the water molecules is artificial and random. Thus, we can not use the detailed approach on the surface part of the poten-

tial. Secondly, the probability of getting e.g. a water-amino acid pair can not be computed using a p_{hoh}, because the number of the water molecules is artificial. Expressions such as $p_{a,hoh} = p_a p_{hoh}$ thus do not make sense. Therefore, we assume that the expected frequency of an amino acid appearing on the surface is equal to the overall frequency of that amino acid. The expected frequency of an amino acid-water pair is simply the observed frequency of the amino acid in the data base:

$$f^{exp}_{a,hoh} \propto f^{obs}_{a} \tag{31}$$

The expected frequency of a triangle between amino acids a and b and a water molecule hoh is given by:

$$f^{exp}_{a,b,hoh} \propto f^{obs}_{a,b} \tag{32}$$

This term was used even when the tetrahedral term was computed using the pair frequencies.
The expected frequency of a triangle containing two waters is:

$$f^{exp}_{a,hoh,hoh} \propto f^{obs}_{a} \tag{33}$$

However, this information seems to be incorporated in the above pair water term already, so we decided not to use it. The expected frequency of a tetrahedron containing water is computed analogously:

$$\begin{aligned} f^{exp}_{a,b,c,hoh} &\propto f^{obs}_{a,b,c} & (34)\\ f^{exp}_{a,b,hoh,hoh} &\propto f^{obs}_{a,b} & (35)\\ f^{exp}_{a,hoh,hoh,hoh} &\propto f^{obs}_{a} & (36) \end{aligned}$$

Again, all Terms using more than one water seem to be redundant and have been neglected.

Neglecting possibly redundant information is also backed by results we obtained using our potential. The z-score increases when either pairs, triangles or tetrahedrons containing water were not used in computing the z-score , but only if this redundant information was used. Ignoring it not only improves the z-score but, as is to be expected, the performance does not decrease if the full information is used.

3.6 Data Base

As data base a subset of the PDB data base [6], Select PDB [34], in the version from April 2002 was used. It provides PDB-files with a sequence identity of less than 25%. Some structures exhibited unusually large distances between the C_βs of neighboring amino acids. These structures have been studied manually to determine whether they should be used for computation of the potential or not. Since all of them showed a chain break where the large distances occurred, they had to be expelled from the data base.
The other proteins were tested with Sippl's `PROSAII`, where some of them showed very bad z-score considering their chain length, as can be seen in figure 10. The five proteins we investigated were

1jb0, photo-system I of cyanobakteria

1qle, a complex between an antibody and a protein

1gk9, a substrate-enzyme complex

1f8e, a protein complex with its inhibitor

1pho, an outer membrane protein.

All of these proteins were expelled from the data base because they are either complexed proteins or membrane proteins, which explains their bad z-score.

Furuichi *et al.* showed that the quality of the data base can bias the quality of the potential, at least concerning statistical pair potentials [26]. Using only all α proteins in the data base will lead to better z-score s for all α proteins and worse z-score s for all other proteins. Because Furuichi suggests

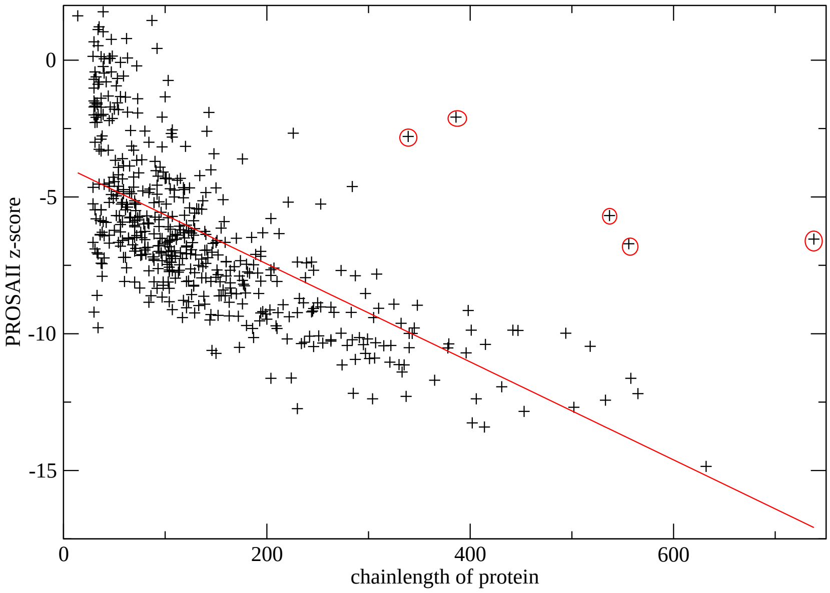


Figure 10: `PROSAII`z-score of the proteins in the select-PDB data base. The red encircled proteins have been deleted.

that mixed data bases will yield the best results on average, we tried to make sure our data base is such a mixed one. Using the SCOP data base [43] we classified the proteins of the PDB Select data base (Table 2). As can be seen, PDB Select contains a mixture of classes of secondary structure.

3.7 Sparse Data

Suppose the data base provides us with N measurements (e.g. N contacts) of which a show the feature we are interested in (e.g. a **A**-**L** contacts). For large N (i.e. a large database) a good estimate for the probability p is obviously $p \approx a/N$, but this will work poorly for small N and rare features (e.g. our potential will diverge if $a = 0$). If we have a prior expectation f^{exp} for the value of p (e.g. the frequency of **A** and **L**, already gives us some information on the frequency of **A**-**L** contacts), then the measured frequency $f^{obs} = a/N$ will be a good estimator for p if $N \gg \frac{1}{f^{exp}}$. If this is not the case, some sparse data correction is in order. Following Sippl [49] and using Bayesian

SCOP class	% of total
α and β	31%
all α	26.5%
all β	21.5%
others	20%

Table 2: SCOP classes of PDB Select proteins. Others include multi-domain, membrane, small and coiled/coil proteins as well as peptides

reasoning, we tried to find a proper estimator for p [35].

Bayesian Inference

Given the true value of p, computing the probability of a certain outcome of our measurement is usually straightforward. I.e. the probability $P(D|p)$ of our data D given p can be written down. Bayes' theorem states that

$$P(D|p)P(p) = P(p|D)P(D)$$

and therefore

$$P(p|D) = \frac{P(D|p)P(p)}{P(D)},$$

which is what we need for computation.
For our purpose the denominator $P(D)$ can be viewed simply as a normalization constant, as it is independent of p; The quantity $P(p)$ is called *prior*. Now there are two problems left to solve: Instead of a simple estimate for p we have to deal with a probability distribution $P(p|D)$. The Maximum Likelihood Principle is used to solve this problem. This leaves the problem of choosing a suitable prior $P(p)$.

The Prior

As stated above, we have some expectation f^{exp} about p, but we need the whole probability distribution. Since we do not know the prior, the best we can do is to construct one that is both reasonable and convenient. For usage of a maximum entropy estimate, we postulate the following properties:

1. The maximum of $P(p)$ should be at f^{exp}, so that we'll estimate $p = f^{exp}$ for $N = 0$ (i.e. no data).

2. p should never be 0 or 1, so that the potentials stay finite. Thus we have $P(0) = P(1) = 0$.

This seems to sum up all desired properties of the prior, so we can choose a convenient form for the prior and use the piecewise linear function

$$P(p) \sim \begin{cases} \frac{p}{f^{exp}}, & p \leq f^{exp} \\ \frac{1-p}{1-f^{exp}}, & p > f^{exp} \end{cases}$$

Maximum Likelihood Estimate of the Parameters

If we assume independence of our N measurements, $P(D|p)$ is a simple binomial distribution $P(D|p) \sim p^a(1-p)^b$, with $b = N - a$. The Maximum Likelihood estimate for p is a value that maximizes $P(p)p^a(1-p)^b$. To find it, we have to distinguish three cases:

The maximum could be at some $p \leq f^{exp}$, in which case it has to fulfill

$$\frac{d}{dp} p \cdot p^a(1-p)^b = 0$$

$$(a+1)p^a(1-p)^b - bp^{a+1}(1-p)^b = 0$$

$$p = \frac{a+1}{N+1}.$$

If $p \geq f^{exp}$, we have

$$\frac{d}{dp}(1-p)p^a(1-p)^b = 0,$$

which eventually yields

$$p = \frac{a}{N+1}.$$

Else the maximum might by at $p = f^{exp}$.

Application of these three cases yields the "observed frequencies" we used at all computations:

$$f^{obs} = \begin{cases} \frac{a+1}{N+1}, & \frac{a+1}{N+1} < f^{exp} \\ \frac{a}{N+1}, & \frac{a}{N+1} < f^{exp} \\ f^{exp}, & \text{else} \end{cases} \tag{37}$$

Because we use $\ln \frac{f^{obs}}{f^{exp}}$ for our potential, the above estimate has the nice property that the energy contributions equal 0 for all cases where not enough data is available.

Depending on the level of detail used in the potential, this occurs in 1% to 20% of all cases.

4 Computational Results

In our computations, we used two different poly-proteins, both derived from poly-proteins used in Sippl's `PROSAII`. One, which we will call "10k", is 9999 amino acids long, the other, called "30k" has a length of 30681 amino acids. If not stated otherwise, the 10k poly-protein has been used for computation.

4.1 Calibrating the Potential

Before testing the potential the variable parameters of the potential have to be calibrated. Basically, there are three distinct questions to be answered:

- What are the best distances in the water grid?
- What is the desired ratio between the energy contribution of the surface and the core?
- How many and what distance classes are there to be?

4.1.1 Calibrating the Water Grid

There are two distances which can be adjusted concerning the water grid:

- The minimal distance of water to the C_β atoms of the amino acids
- The distance between the virtual water molecules in the grid

It is easily understood that the minimal distance to the amino acids has to be as small as possible while ensuring that there is no water misplaced inside the protein. We compared the number of water molecules added when using C_β only with the number of waters added when using all atoms. If these two numbers are equal, we can be sure that no water is placed into erroneous cavities. We found that a minimum distance of 5.2Å fulfills this requirement. This quite big minimum distance also renders it all but impossible to detect cavities inside the protein, an undesirable effect that could only be avoided by always using side chain coordinates also for the water shell creation. However,

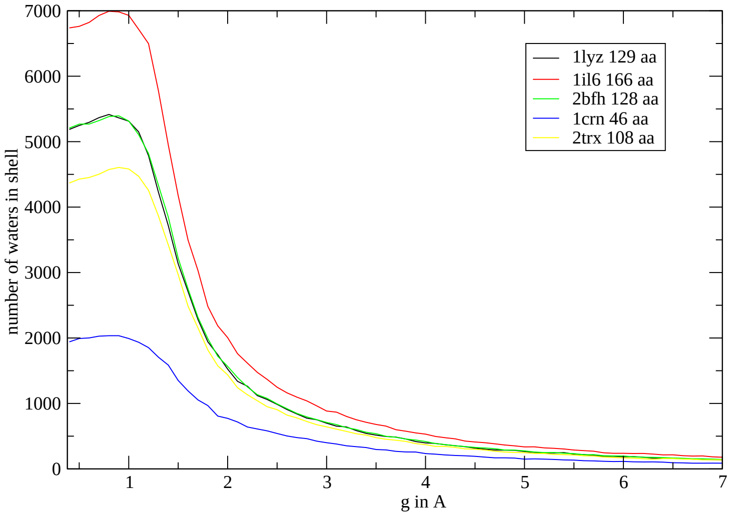


Figure 11: Number of water molecules added as a function of the distance in the water grid. Different proteins with different length were used

this would lead to problems concerning the location of the side chains in the poly-proteins used during the threading procedure.

Concerning the grid distances of water, it would be preferable to have as small a distance as possible because this would give an excellent resolution of the surface. But the number of points to be tessellated rises with g^3, which makes it necessary to find a compromise between the number of points added and the resolution of the surface. It is important to point out that the number of points added in a cubic grid corresponds to the probability that because of the cubic nature of the grid and in spite of the random dislocation added the tessellation will fail due to numerical problems of the `qhull` library. This will result in even more time needed for the computation because the tessellation has to be repeated. Basically, we decided to leave the distance of the water grid to be changeable by a command line option, and set the default to 5.1 Å, which seems to be a good compromise. As is shown in Figure 11, using a bigger grid distance will not significantly decrease water molecules added before tessellation.

4.1.2 Calibrating the Scale of the Surface Term

After introducing the water grid, the energy contributions have to be put into relation to the energy contributions of the amino acid pairs. There are two different problems to be solved. One concerns the absolute weight of the surface term, the other concerns the necessity to make the absolute contribution as independent of the grid distance as possible.
The latter question was solved by fitting a simple function into the graph of number of water pairs to grid distance, as shown in Figure 12.

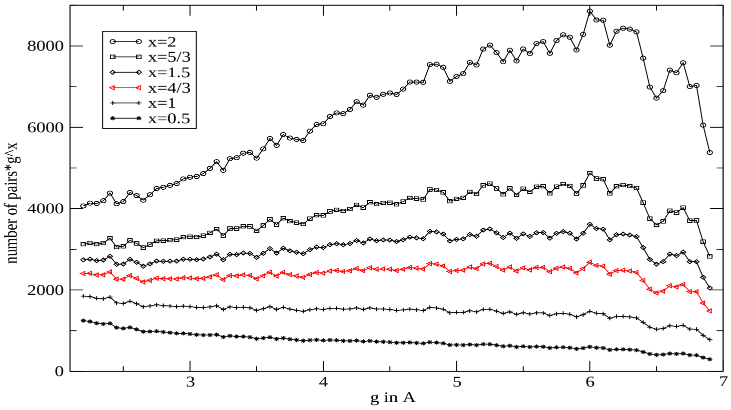


Figure 12: Number of Pairs between amino acids and water multiplied by a scaling function $f(g)$ for crambin (1ab1) and $d_{min} = 4.2$Å. As can be seen, the best fit (red) is $g^{\frac{4}{3}}$

The best connection is not a quadratic one but $g^{\frac{4}{3}}$, which is due to the more complicated interactions between the surface of the protein and the water grid as a result of the tessellation.
By applying this correction we can achieve a z-score almost independent of the distance of the water grid (Figure 13).

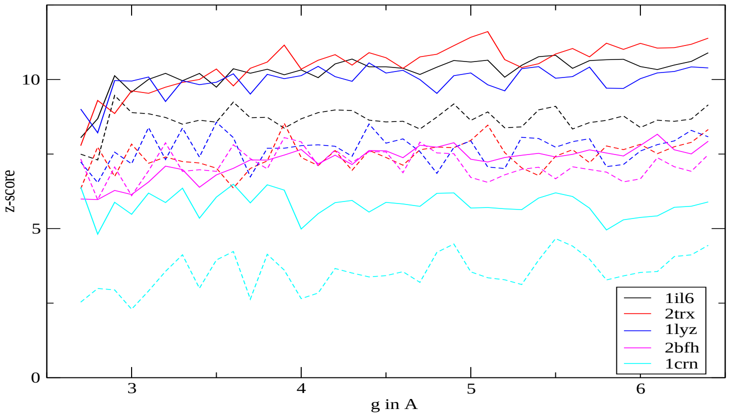


Figure 13: z-score of 5 different proteins as a function of the distance of the water grid in Å. Full line is total z-score, dotted line z-score of the surface term only

The absolute weight of the surface term was chosen so that the energy contribution of the surface term is roughly the same order of magnitude as the contribution of the core term. A factor of 0.05 seems to achieve this.

4.1.3 Calibrating the Sequence and Euclidean Distance Classes

Two different types of parameters can be adjusted concerning the distance classes, being the number of classes and the boundaries thereof. In calibrating the distance classes, it is necessary to remember the necessity of sufficient data for relevant statistics, so too many classes or classes too small are not advisable. The distribution of the euclidean distances can be seen in Figure 14. The boundaries of the euclidean distance classes are chosen as shown in Table (3).

Of course because of steric reasons there are only very few atom pairs with an euclidean distance of 0-1 Å, and C_β C_β distances of under 3 Å are rare indeed, with a total occurrence of less than 0.2 %. Also, the nature of the tessellation assures that there are very few $C_\beta - C_\beta$ pairs with a distance bigger than 10Å.

Number of classes	class 1	class 2	class3
1	0-∞		
2	0-7Å	7Å-∞	
3	0-4Å	4-7Å	7Å- ∞

Table 3: Boundaries of euclidean distance classes

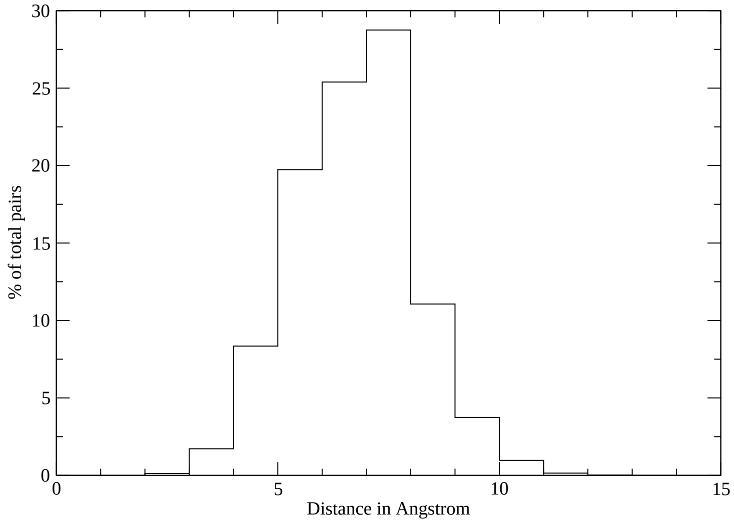


Figure 14: Distribution of euclidean distances in classes of 1Å size. 287610 total pairs used.

The boundaries of sequence distance classes are shown in Table (4).

An interesting observation is that obviously every possible contact with a sequence separation of 1 exists (Figure 15). Since there seems to be no information to be gained by using them, it should be possible to simply remove these pairs out of the computation without a loss of performance. We did not do this for two reasons. Firstly, the computational effort saved is about the same as the effort for the removing, and secondly, while very small indeed, the chance to get such a pair with a distance separation of more than 7 Å in

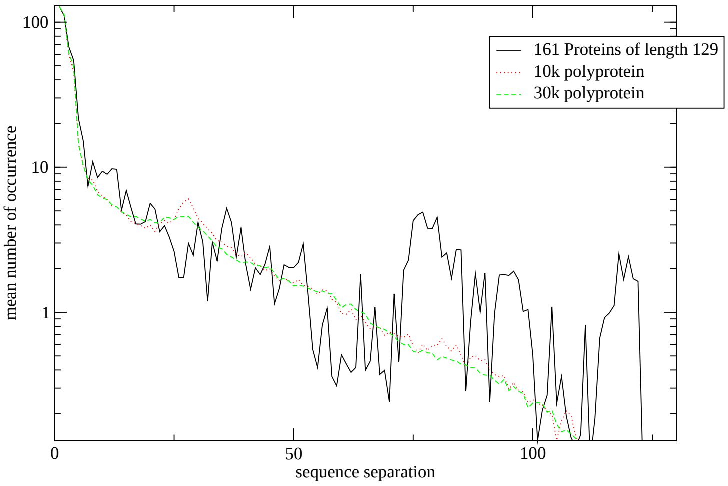


Figure 15: Number of pairs against sequence separation for proteins of 129 amino acids length. Note that every possible pair (128 per protein) with distance 1 exists.

# of classes	class 1	class 2	class3	class4	class5	class6	class7
1	1-∞						
2	1-5	6-∞					
3	1-5	6-10	11-∞				
4	1-10	11-50	51-90	91-∞			
5	1-5	6-30	31-60	60-90	91-∞		
6	1-5	6-10	11-40	41-70	71-100	101-∞	
7	1-5	6-15	16-30	31-50	51-75	76-100	101-∞

Table 4: Boundaries of sequence distance classes

the poly-proteins is about 10 times bigger than in native proteins.

When comparing the performance of the potential using different numbers of euclidean and sequence distance classes (Figure 16), we can see that the performance increases when using two euclidean distance classes, but more sequence classes do not linearly improve the performance. For deciding upon

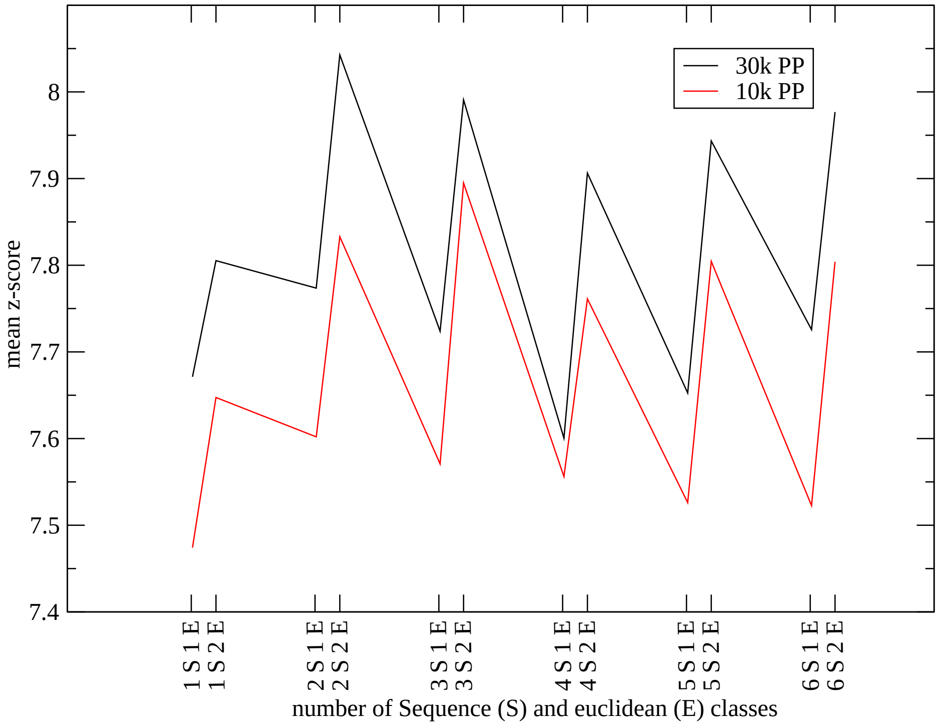


Figure 16: Mean value of z-score of 5 different proteins: 2trx, 2bfh, 1il6, 1ab1, 1lyz. Two different poly-proteins, 10k and 30k, as well as different numbers of sequence and euclidean distance classes are used. 1S1E means that one sequence and one euclidean distance class are used for the computation.

PDB code	length of protein	Weberndorfer's	`SaDSaT` 30k	`SaDSaT` 10k	`PROSAII`
1il6	166 aa	10.195	9.2998	9.2005	-8.21
2bfh	128 aa	6.960	7.1567	6.818	-6.53
1lyz	129 aa	9.430	8.7682	8.7769	-9.16
1ab1	46 aa	5.483	5.7547	5.6863	-5.54
2trx	108 aa	9.247	10.7027	10.583	-9.27

Table 5: Z-scores of different proteins using different potentials

which combination of distance classes to use for computation, we had to use the performance on decoy sets, too (Table 8 in chapter 4.2.2).

The combination of the informations lead us to use 5 sequence and 2 euclidean distance classes for our computations. So when not stated otherwise, all results below are computed using this combination of sequence and euclidean distance classes.

4.2 Performance of the Potential

4.2.1 Z-scores

Directly comparing z-scores is of course of only limited value for comparison of knowledge based potentials. Thermodynamic studies have shown that for a protein of 100 amino acids in size, the z-score should range about or higher than 15 [65]. These z-scores can not be achieved with this potential, neither with any of the others tested as of yet. However, the new potential should not score too badly compared to others, so tests were calculated.

The results (Table 5) seem to indicate that all three potentials are about equally successful, but using different, and better, indicators for the quality of the potentials soon destroys this equality (Chapter (4.2.2).

It is interesting to see that usage of a bigger poly-protein does not increase the quality of the potential much. Another interesting fact is while the 30k poly-protein is roughly 3 times bigger as the 10k, the tessellation takes about 8 times longer, even when there are no errors occurring during tessellation. The size of the proteins to be tessellated is independent of the length of the poly-protein, only the number of tessellations to do increases. The fact that more than 3 times more time is needed for the bigger poly-protein seems to be due to the time it takes `qhull` to perform this tessellation, which is not always only dependent on the number of points tessellated. The authors state that the best case performance is $\mathcal{O}(n \log n)$. There seem to be many more cases where it is more costly to compute the tessellation in the 30k poly-protein.

Another test of a potential's performance is its ability to discern between the right sequence to a structure and wrong ones. As can be seen in Table (6) we used 6 different proteins with a chain length of 129 amino acids to test the potential. In all cases but two, the z-score of the native sequence on the native structure is the best by far. The exceptions are the proteins 1uih and 135l, lysozymes from hen and turkey, respectively. This two proteins a virtually impossible to discern, as their 3d-structures are almost congruent, having a C_α rmsd of 0.488Å. Furthermore, the sequence identity exceeds 90%. It is therefor easily explicable that the sequence of 1uih yields a slightly better z-score on the 135l structure than the native sequence does. For all the other proteins, which are not related, `SaDSaT` can discern between the native sequence and a natural sequence folding to another structure.

4.2.2 Decoy Sets

One method to test the performance of a potential for protein folding is using "decoy" sets, where physically feasible non native structures are created specifically for testing the ability of potentials to find the native structure in a set of artificial structures designed to fool it. We used the decoy set constructed by Park and Levitt [44], a set of 7 proteins with an average of 665 decoys per protein. As can be seen in Table 7, Sippl's `PROSAII` performed best

sequence	1uih	1knm	1dyz	1ahm	135l	2cyk
structure						
1uih	**9.128**	-0.420	-0.096	2.306	**8.886**	0.403
1knm	-0.318	**5.086**	-0.050	-1.075	0.242	-0.440
1dyz	-0.864	-1.273	**7.220**	-1.116	-0.740	0.262
1ahm	-0.212	-1.947	1.499	**9.259**	0.205	-1.187
135l	**9.091**	-0.324	-0.231	1.852	**9.042**	0.707
2cyk	-0.132	0.721	-0.130	-1.030	-0.002	**8.765**

Table 6: Performance of the `SaDSaT` potential on different proteins with length 129, the proteins are 1uih, hen lysozyme, 1knm, streptomyces hydrolase (complexed with lactose), 1dyz, alcaligens electron transport, 1ahm allergen of dermatophagoides, 135l turkey lysozyme and 2cyk human cytokine. 1uih and 135l are almost identical lysozymes.

PDB code	`PROSAII` ranking	RMSD	`SaDSaT` 1 ranking	RMSD	`SaDSaT` 2 ranking	RMSD	Weberndorfer's ranking	RMSD
1ctf	1	1.819	1	4.096	1	4.069	2	1.445
1r69	1	1.663	4	1.663	12	1.663	7	4.760
1sn3	1	5.395	1	7.292	1	2.221	40	6.868
2cro	1	1.093	10	2.045	26	2.700	4	2.032
3icb	1	1.872	2	1.408	7	1.408	14	2.516
4pti	1	1.485	3	4.998	3	4.693	10	2.590
4rxn	8	2.104	3	2.351	3	2.044	19	2.104

Table 7: Comparison of four potentials using the Park and Levitt decoy set. RMSD is the root mean square deviation of the lowest ranking decoy in Å, `SaDSaT` 1 is using triangles, `SaDSaT` 2 is not.

by far in this decoy set. When using tessellation potentials, Weberndorfer's Potential did not rank any of the native structures on first place, while both the `SaDSaT` using triangles and the other scored two hits. The performance when using triangles is a little better than without. The correlation between the root mean square deviation, as a measure of the difference between two structures, shows interesting results (Figure 17). It is of course necessary to compare the z-scores, therefore the `PROSAII` z-scores have been multiplied by -1 to be comparable to the `SaDSaT` z-scores. Even in the 2cro decoy set, where the `SaDSaT` force fields show their worst performance concerning the rank of the native structure, the correlation between z-score and the root mean square deviation is better than that of `PROSAII`. This indicates that this potential is potentially useful for theoretical applications like works about the folding space of proteins, where it is important to have fast computational methods which will not necessarily be used for natively folded proteins.

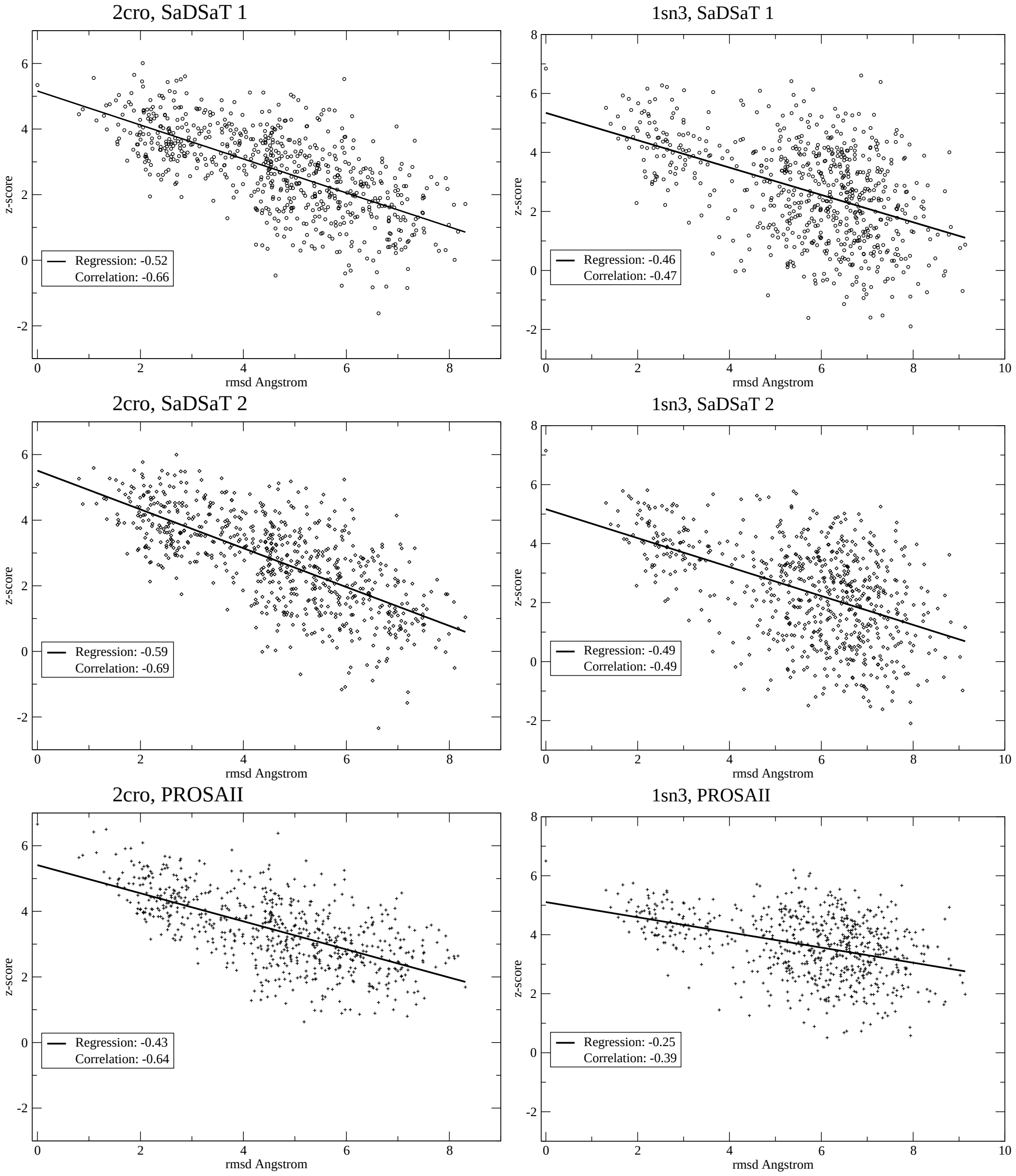


Figure 17: Correlation between the root mean square deviation (rmsd) of the decoy sets and the z-score . `SaDSaT` 1 is not using triangles, `SaDSaT` 2 is. In both examples, `SaDSaT` shows better correlation than `PROSAII`

PDB code	rank 11	rank 12	rank 21	rank 22	rank 31	rank 32
1ctf	5	**1**	3	2	3	2
1r69	**4**	**4**	5	8	7	5
1sn3	1	1	1	1	1	1
2cro	28	**14**	19	16	29	28
3icb	15	8	9	16	**7**	**7**
4pti	1	1	3	1	1	1
4rxn	7	10	10	5	8	**4**
mean	8.714	**5.571**	7.142	7	8	6.857

PDB code	rank 41	rank 42	rank 51	rank 52	rank 61	rank 62
1ctf	5	2	2	**1**	2	2
1r69	**4**	6	6	6	6	5
1sn3	1	1	1	1	1	1
2cro	18	16	20	15	26	19
3icb	9	10	9	10	**7**	**7**
4pti	3	1	3	1	1	1
4rxn	12	9	10	6	10	7
mean	7.429	6.428	7.286	5.714	7.571	6

Table 8: Ranking of decoy sets with different number of euclidean and sequence distance classes. Best performance **bold**

4.2.3 Stability Changes

Another indicator for the quality of a potential is the quality of correlation between experimentally derived energy changes due to mutations and the differences in energy computed for these mutations with the potential.
Following Carter *et al.* [16], we used the same proteins with the same mutations to compare the performance of their potential, a four point contact potential using the center of mass to represent the amino acids, with ours. The proteins used were 1ey0, 1b2x, 1ci2, 1l63 and 1igv (Figure 18).
Since all mutations were hydrophobic core mutations, it is not surprising that the correlation was better most of the time when not using the surface term. While our potential shows a comparable or even slightly better performance at 1b2x and 2ci2, the other proteins and the overall performance are clearly in favor of Carter's potential. This could be due to the usage of the center of mass for representing the amino acids, which is bound to give a better representation of the amino acids, but can not be used easily for the threading procedure.

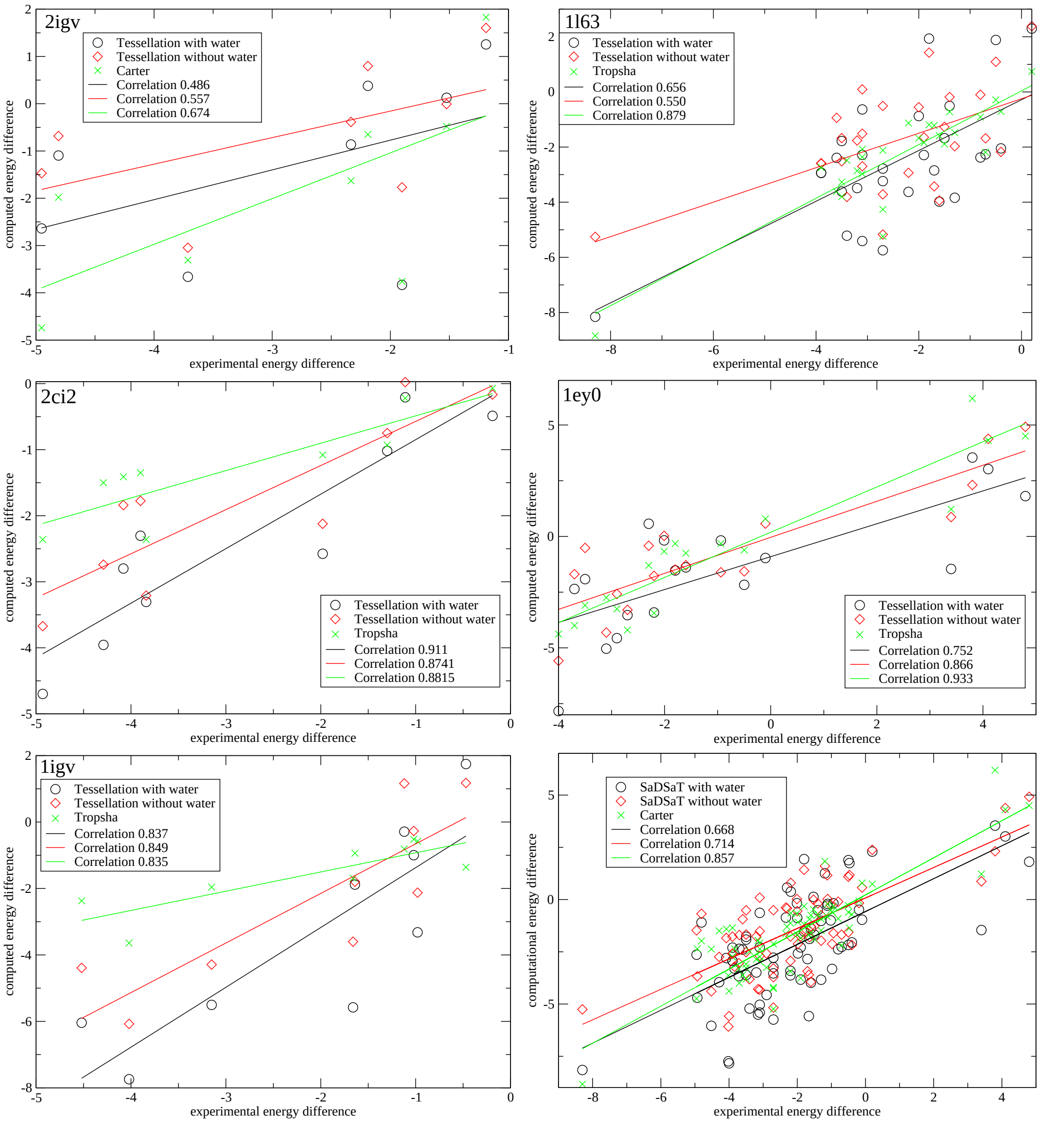


Figure 18: Correlation between experimental $\Delta\Delta G$ and computed energy difference for 2igv, 1l63, 2ci2, 1ey0, 1igv and all together

4.3 Inverse Folding

Inverse folding is trying to find a sequence which will fold into a given structure. One means to achieve this is using adaptive walks. Starting with the target structure and a random sequence, one tries to find better sequences by mutating single amino acids and keeping the mutation in question if and only if it results in a better z-score of the sequence on the target structure.

There are different ways of mutating an amino acid which differ in the probability to get a certain amino acid. These are mainly:

- a totally random approach, where the probability to get an amino acid is depend end on the frequency of the amino acids only
- a surface/core approach, where the probability to get an amino acid depends on the location in question and
- an approach dependent on the secondary structure and the environment at the location of the mutation following Bowie *et al.* [12].

The adaptive walk is terminated if no further mutation can be found which leads to a better z-score . In our case, this was defined as mutating the sequence 100 times without improving it. In order to get information about the quality of the adaptive walk, we computed the z-score of all the sequences generated using `PROSAII` and compared them with the z-scores of our potential. The longer a linear correlation between the two z-scores is, the better the potential in question. The end of the linear correlation indicates an improvement of the energy which is not due to physical reasons but due to specialties of the potential, in other words the sequence is adapted to the special properties of the potential.
As can be seen in Figure 19 below, the only effect of using bigger poly-proteins for the threading procedure is a slightly steeper descend. However, this has to be bought by higher computational costs. Therefore, it is highly uneconomic to use a bigger poly-protein for adaptive walks.

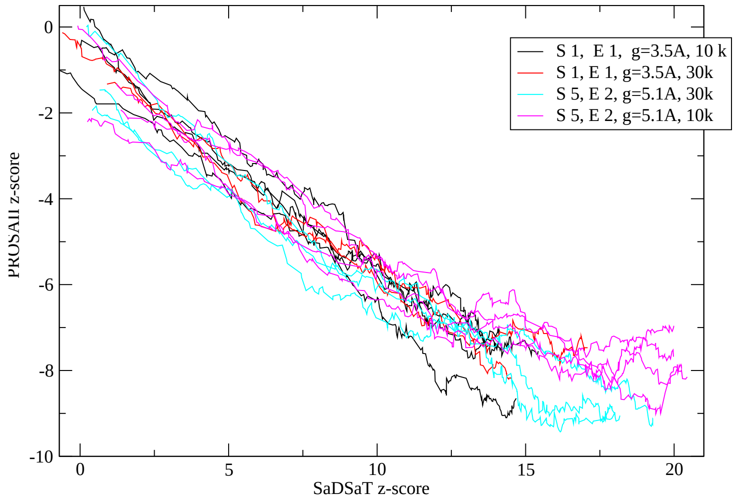


Figure 19: Comparison between adaptive walks using a 10k and adaptive walks using a 30k poly-protein, respectively.

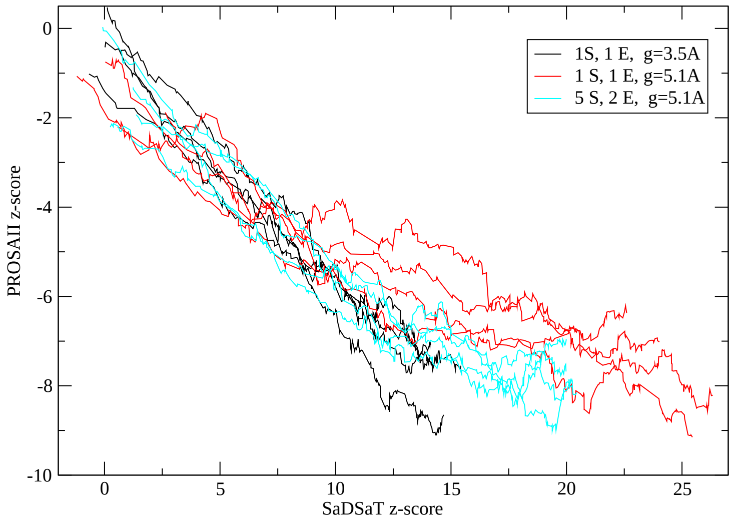


Figure 20: Adaptive walks using different number of sequence and euclidean classes and a 10k amino acid poly-protein.Only the approach with one class and a big grid distance is significantly worse than the others.

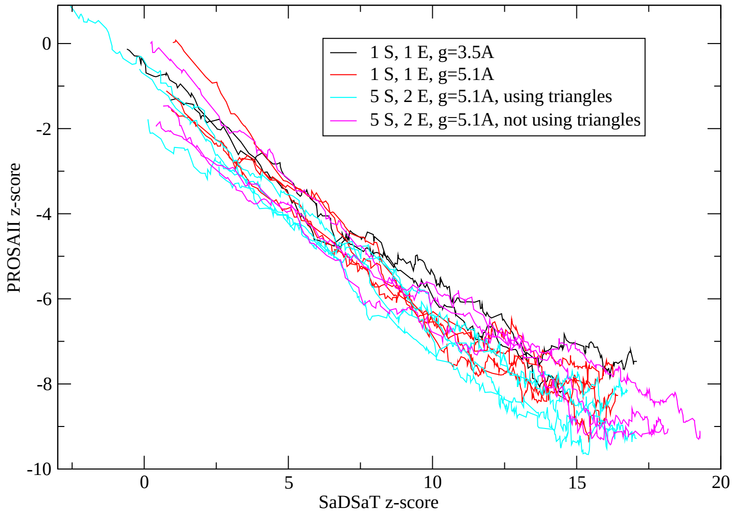


Figure 21: Adaptive walks using different number of sequence and euclidean classes and a 30k poly-protein.

Figures 20 and 21 show that using a more detailed approach does not really improve the performance of `SaDSaT` at adaptive walks. The performance when using only one class i.e. when not using any of Sippl's sequence and euclidean classes, is comparable to that of `SaDSaT` when using high details, if g is sufficiently small.

This would indicate that using the mergence of Sippl's and Tropsha's approaches is in fact not useful.

But comparison with adaptive walks computed with Weberndorfer's potential show a different result (Figure 22)

`SaDSaT` performs better than Weberndorfer's potential does, but it remains to be seen whether the increase in quality is not bought by a too high increase in computational effort, and whether it is due to the usage of sequence classes or the usage of a better surface term.

Another important feature of an adaptive walk is the number of steps it needs to get to the end. Obviously, the faster that happens, the better. For using

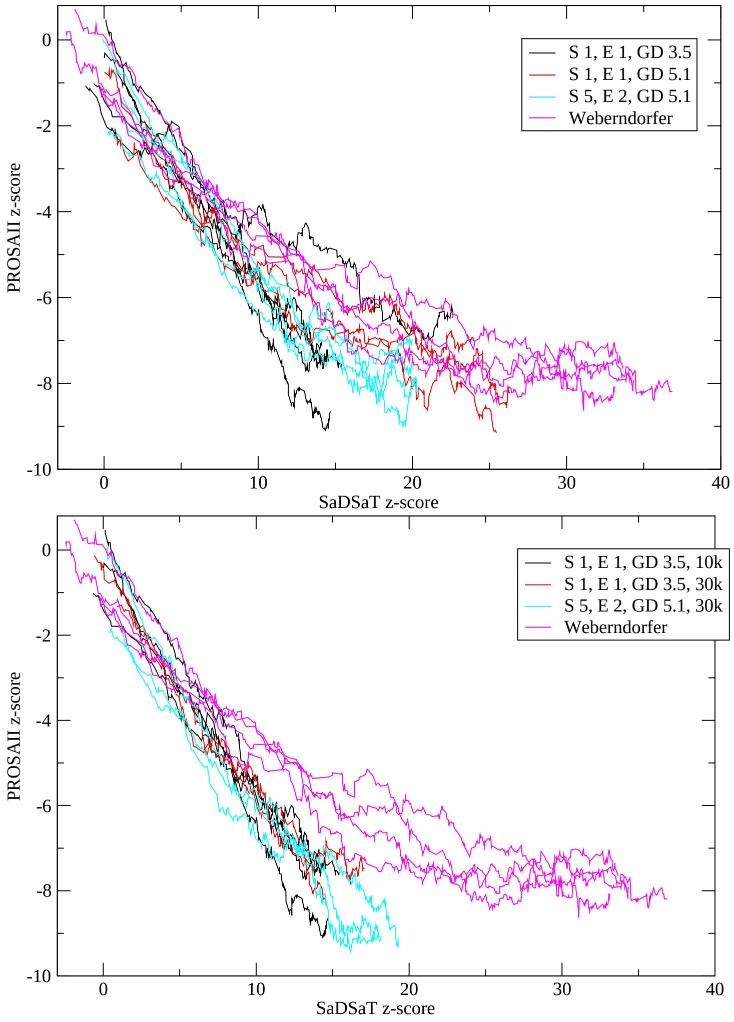


Figure 22: Comparison of different adaptive walks using `SaDSaT` with adaptive walks using Weberndorfer's potential. It can be seen that Weberndorfer's potential is outperformed by `SaDSaT`

the 10k poly-protein and 2bfh, the mean number of steps `SaDSaT` needs to finish an adaptive walk is 276.1, for the 30k poly-protein it is 286.3, while Weberndorfer's potential needs a mean of 394.5 steps to finish. This indicates that the 10k poly-protein with `SaDSaT` is the potential of choice out of this three, as it is computed faster than the 30k variant, needs fewer steps than both and has better quality then Weberndorfer's.

Comparing `SaDSaT` generated adaptive walks with adaptive walks generated by `PROSAII` it can be seen that both potentials succeed in exceeding the native z-score of the other potential (see Figure 23). However, to reach the native z-score of the potential used to evaluate the adaptive walks, both `SaDSaT` and `PROSAII` have to reach values well over their respective native z-score . Ideally, one would hope to reach both z-scores approximately at the same time. While `PROSAII` performs a little better in absolute z-score values reached, it takes longer to terminate the adaptive walk. The average length of the adaptive walks computed using `PROSAII` is 433, while `SaDSaT` stays well under 300 (see above).

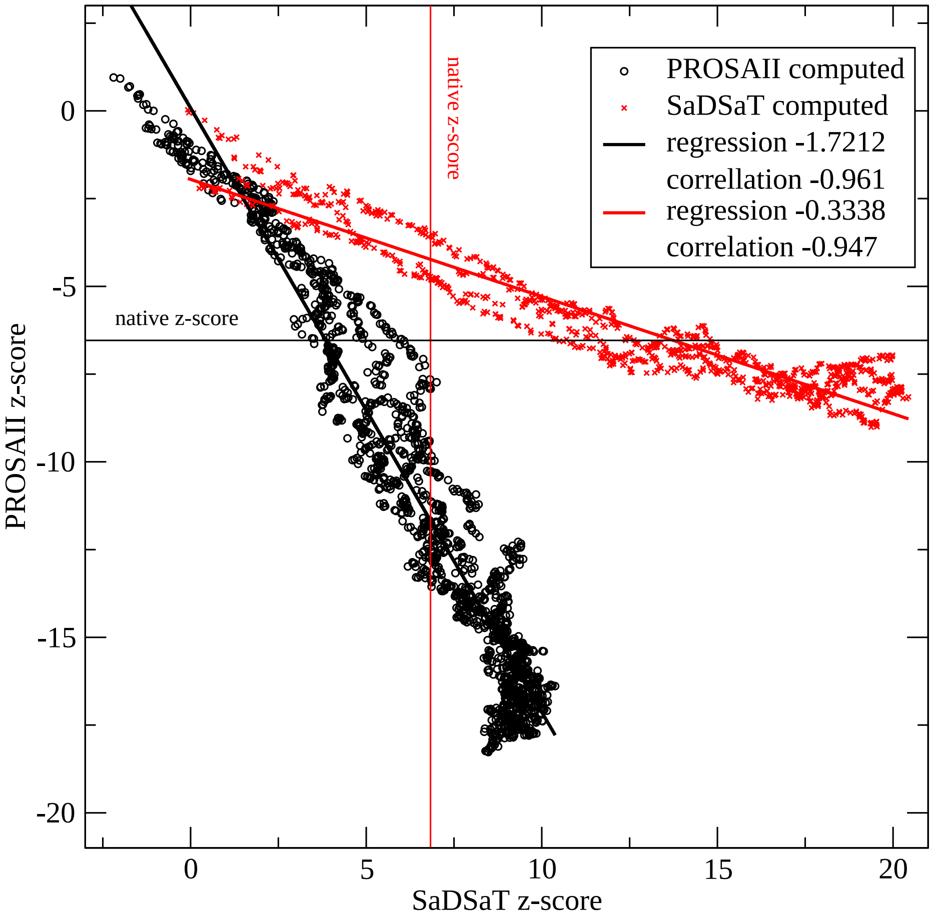


Figure 23: Comparison of adaptive walks using `PROSAII` and `SaDSaT` 4 adaptive walks on 2bfh were computed with each potential.

4.4 Computational Effort

Most of the computational effort of the program is caused by `qhull` in performing the Delaunay tessellation. Because of the water shell added to the problem, `SaDSaT` needs more computational time there than Weberndorfer's potential does. The `qhull` algorithm has a best case performance of $\mathcal{O}(n \log n)$. As stated earlier, the usage of the smaller (10k) poly-protein seems to be sufficient for acceptable results, while keeping the memory needs relatively low and the processor time small.
On a Pentium III 1GHz system the following times are needed to compute the z-score of a protein 129 amino acids long. `SaDSaT` without the triangle term was used. For using the 30k poly-protein we need:

> 153 min and 706MB RAM if poly-protein tessellation is computed.
>
> 33 seconds and 1046MB RAM if poly-protein tessellation is read.

and for using the 10k poly-protein:

> 12 min and 241MB RAM if poly-protein tessellation is computed.
>
> 11 seconds and 338MB RAM if poly-protein tessellation is read from disk.
>
> 6.9 seconds if tessellation is in memory and a new sequence is put in.

The ratio of the time needed to compute the tessellation and the time needed for z-score computation exclusively is about 103:1. If the 30k poly-protein is used, it is even bigger, exceeding 250:1. This big difference shows that `SaDSaT` is better suited to applications like inverse folding than to repeatedly computing z-scores for different structures. If the structure to be evaluated stays the same, the step determining the computational effort is the re-computation of the z-score, which takes about 7 seconds in the example above. The same is true for decoy set experiments, where there is no need to compute a z-score as the sequence stays the same and it is sufficient to rank the energies themselves.

As indicated above, the usage of the 30k poly-protein does not significantly increase the performance but increases the computational effort at least by a factor 3, even 12 if the tessellation has to be computed.

PDB code	IPF `SadSaT` ranking	RMSD	IPF`SaDSaT` CC ranking	RMSD	`SaDSaT` 2 ranking	RMSD	Weberndorfer's ranking	RMSD
1ctf	2	1.819	1	1.819	1	4.069	2	1.445
1r69	3	1.663	3	1.663	12	1.663	7	4.760
1sn3	1	6.868	1	5.395	1	2.221	40	6.868
2cro	37	2.045	24	2.045	26	2.700	4	2.032
3icb	6	1.408	9	1.408	7	1.408	14	2.516
4pti	11	4.693	9	4.693	3	4.693	10	2.590
4rxn	6	5.682	6	2.104	3	2.044	19	2.104

Table 9: Comparison of four potentials using the Park and Levitt decoy set. RMSD is the root mean square deviation of the lowest ranking decoy in Å, IPF are using Iterative Proportional Fitting, CC is using Contact Capacity, `SaDSaT` 2 is the old one.

4.5 Iterative Proportional Fitting

Using Iterative Proportional fitting for calculating expected frequencies yields higher Entropy and better agreement with the boundary conditions than using `donlp`.
It is possible not only to calculate the expected frequencies of tetrahedrons dependent on pairs, but also on triangles. Thus we have the possibility to compute a `SaDSaT` potential using triangles.

4.5.1 Results using IPF

for comparison the same using the old `SaDSaT` and `PROSA`:

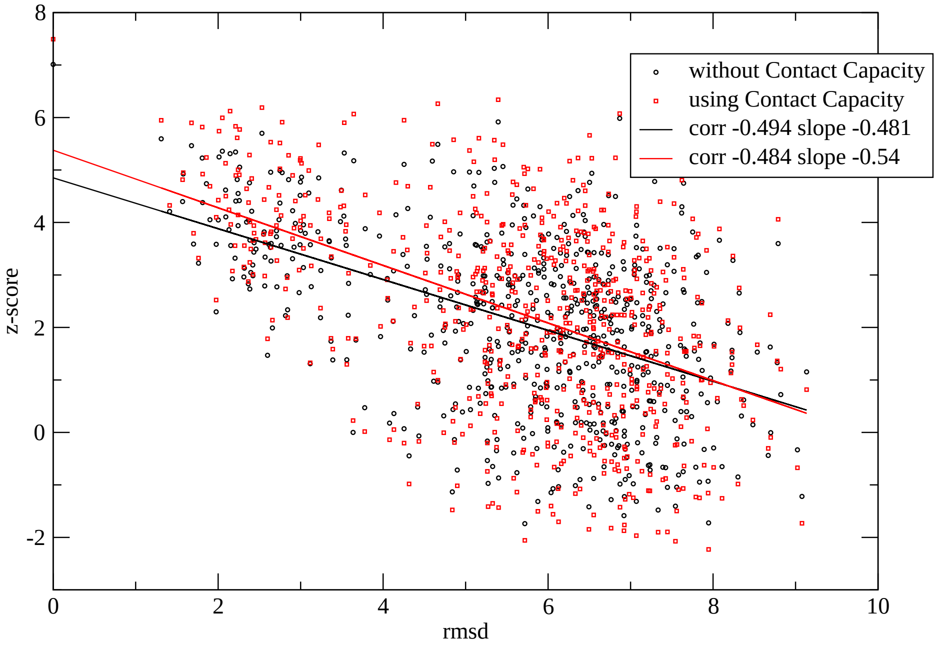


Figure 24: IPF `SaDSaT` rmsd against z-score of Decoy set 1sn3

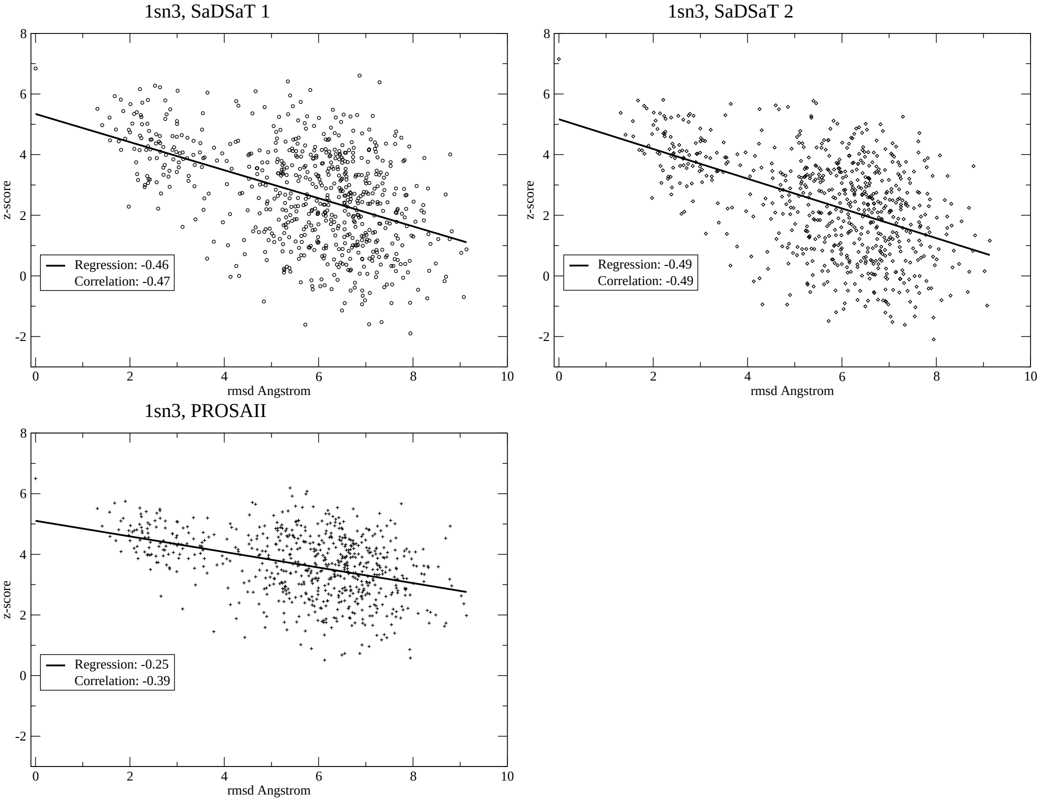


Figure 25: Correlation between the root mean square deviation (rmsd) of the decoy sets and the z-score . `SaDSaT` 1 is not using triangles, `SaDSaT` 2 is. In both examples, `SaDSaT` shows better correlation than `PROSAII`

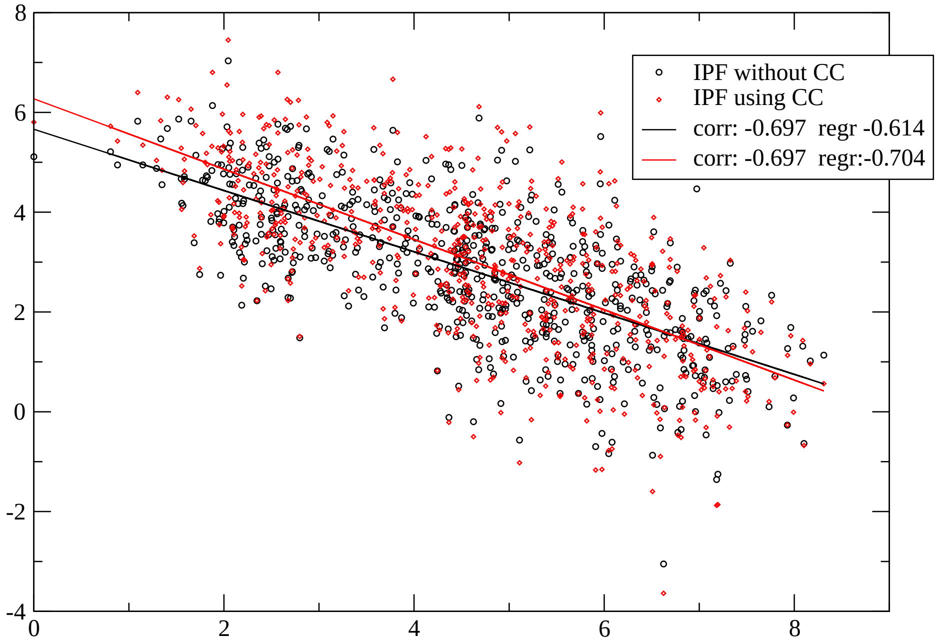
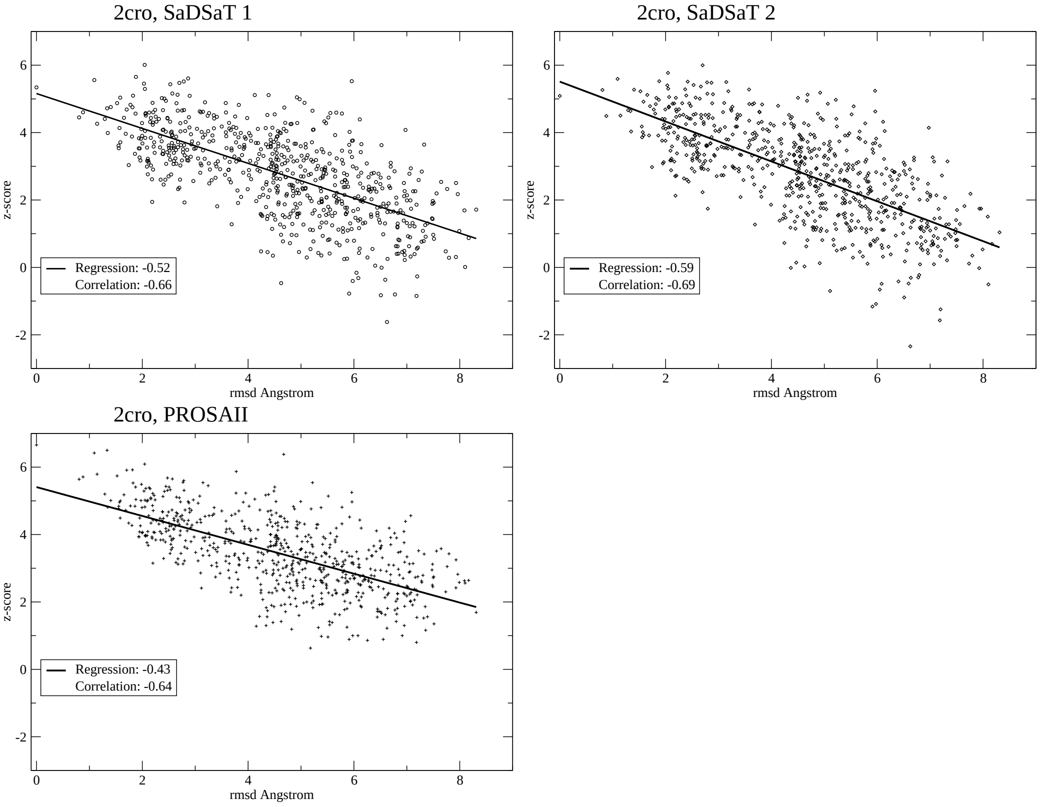


Figure 26: IPF `SaDSaT` rmsd against z-score of Decoy set 2cro

And for 2cro: for comparison the same using the old `SaDSaT` and `PROSA`:


2cro, SaDSaT 1

2cro, SaDSaT 2

2cro, PROSAII

Figure 27: Correlation between the root mean square deviation (rmsd) of the decoy sets and the z-score . `SaDSaT` 1 is not using triangles, `SaDSaT` 2 is. In both examples, `SaDSaT` shows better correlation than `PROSAII`

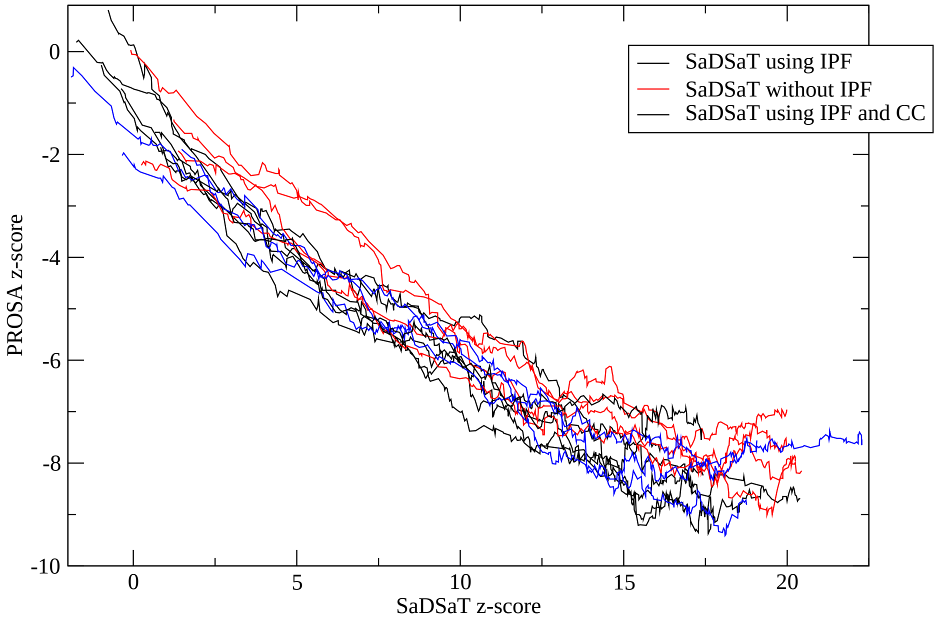
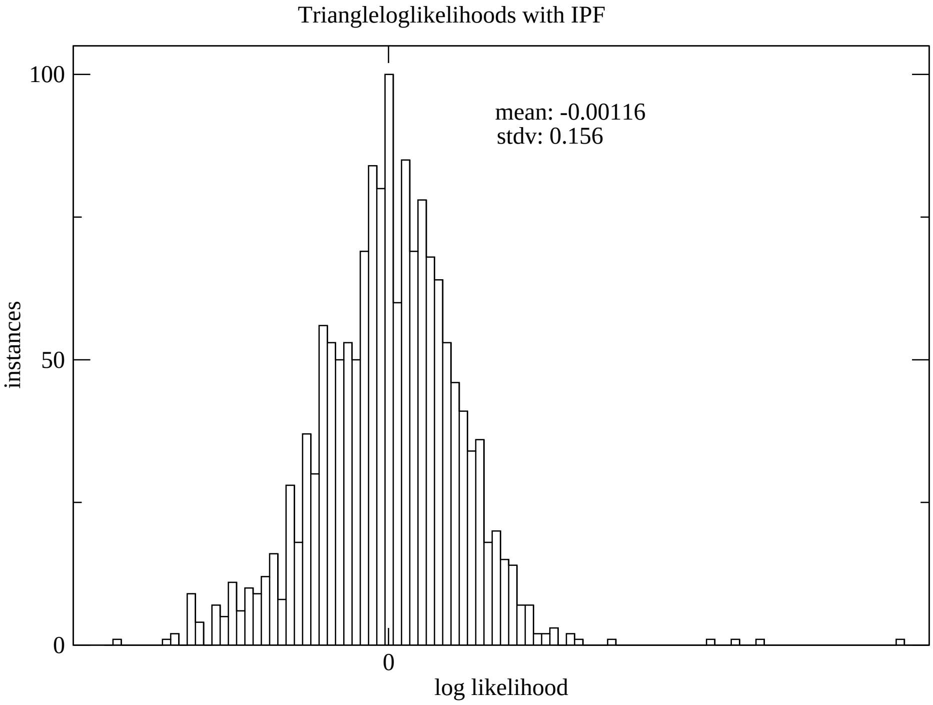
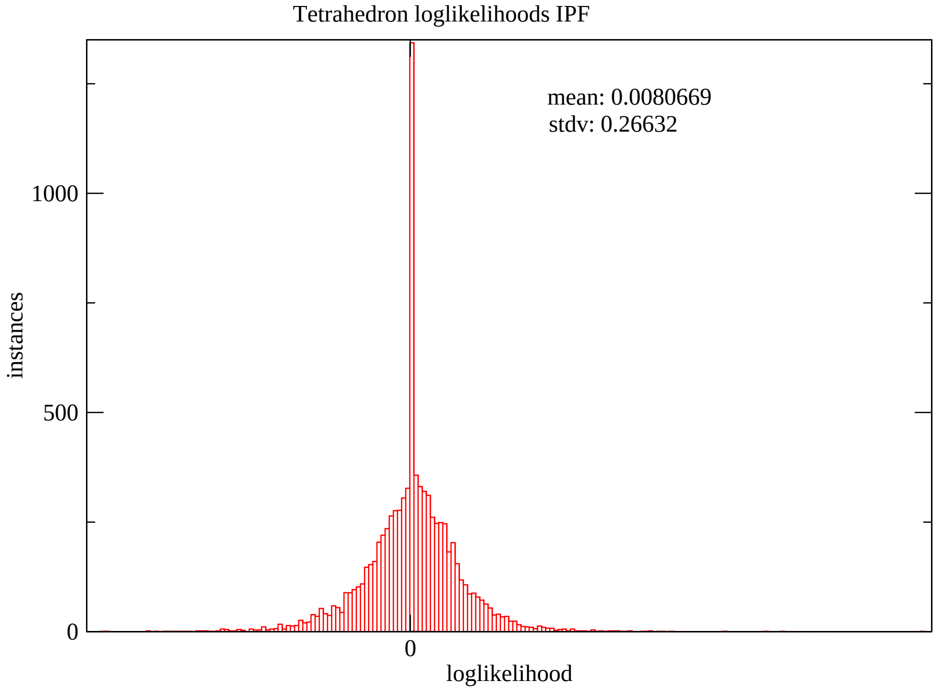


Figure 28: Comparison of adaptive walks using `SaDSaT` with IPF, with CC and without CC, and the old `SaDSaT`.

Performing Adaptive Walks yields the following result against `PROSA` : Termination is achieved after 258.75 steps with the old SaDSaT, after 260,6 steps with the IPF `SaDSaT` without Contact Capacity, and after 278.6 steps using Contact Capacity.


Triangleloglikelihoods with IPF
mean: -0.00116
stdv: 0.156
100
50
0
instances
0
log likelihood


Tetrahedron loglikelihoods IPF
mean: 0.0080669
stdv: 0.26632
1000
500
0
instances
0
loglikelihood

5 Discussion

Tessellation based potentials include higher order information in a natural way. However, until now they did not include distance and separation information. Distance and separation of amino acids are important descriptors of protein structure. To increase the performance of Tessellation based potentials, we therefore developed a four point potential including this information. To be able to do this, the four point potential has been split int second, third and fourth order terms. This raised the problem of expressing higher order information in terms of lower order, which could only be solved numerically. In addition, a better approximation of protein-surface interaction was introduced using an explicit shell of water. The developed potential, called `SaDSaT`, will soon be publicly available as a library of ANSI-C functions and as stand-alone programs at our home-page[3].

To test the quality of this new knowledge based potential, several different computer experiments have been conducted using `SaDSaT`. The experimental results were compared with other implementations of knowledge based potentials. These were Manfred Sippl's `PROSAII` potential, which is distance and separation dependent, and Günther Weberndorfer's tessellation based potential, which is essentially an improved version of Alexander Tropsha's original tessellation potential.

The experiments showed that the application of distance and separation to higher order potentials improves their performance. Adaptive walks conducted using `SaDSaT` have a better correlation to `PROSAII` than Weberndorfer's potential. This indicates a better correlation to the physical properties of protein structure. Nevertheless, adaptive walks conducted with `PROSAII` and evaluated with `SaDSaT` show that both can still be improved.

Testing the performance of knowledge based potentials using decoy sets is especially significant since these sets have been constructed for this purpose.

---

[3] `http://www.tbi.univie.ac.at/`

`SaDSaT` shows better correlation between energy and root mean square deviation from the native fold than `PROSAII` does. This makes `SaDSaT` a good tool for theoretical applications concerning the sequence and structure space of proteins. However `PROSAII` is better than `SaDSaT` in ranking the native structure as structure of minimum energy. `SaDSaT` outperforms Weberndorfer's potential at this task, but further improving the performance is desirable.

Many possibilities to increase the performance of the potential can be thought of. The mathematical problem of expressing the expected frequencies of the tetrahedrons in terms of frequencies of lower order contacts has not been solved. This is unfortunate because we may lose information. Iterative proportional fitting may be the answer to this problem, but it remains to be seen whether it can deal with the high number of marginals needed for this kind of computation.

Another point of interest is the usage of C_β as descriptor for amino acids. If problems with the threading procedure can be circumvented, usage of the center of mass as descriptor could be able to improve the performance, as it seems to be a better descriptor of protein $3d$ structure.

There also could be room for improvement in the exact boundaries of the distance classes, as well as in using a relative definition of sequence distance classes, i.e. defining "long range" contacts as contacts of the N-terminus with the C-terminus of a protein. For now, short proteins can only have short distance contacts, so a higher resolution of distance classes may be needed. Another possibility is to handle amino acids on the termini differently. This is already used to increase the performance of modern protein secondary structure prediction algorithms.

Another approach to improve the potential will be to distinguish tetrahedrons of different shape and size. This should be doable with the amount of data available today. In a recent publication by Krishnamoorthy and Tropsha [39] five different classes of tetrahedrons – depending on whether sequential neighbors take part in the tetrahedron – are used with good results.

All the points stated above will certainly increase the amount of data needed to compute the potential. However, our sparse data correction shows that there is still insufficient data for almost 10% of the possible tetrahedrons. Thus `SaDSaT` as it is today will also improve if the number of experimentally derived 3d-structures of new proteins is increased.
Finally, a different tessellation algorithm specifically tailored for the task of handling the explicit solvent component of `SaDSaT` may be able to greatly decrease computational costs.

As `SaDSaT` is now inferior to many distance based empirical potentials in terms of fold recognition, it may be of value for protein folding applications only after improving it using the ideas stated above. For theoretical applications, especially for exploring the protein space, searching for neutral networks and structural neighbors, it is already well suited.

References

[1] N.N. Alexandrov and N. Go. Biological meaning, statistical significance, and classification of local spatial similarities in nonhomologous proteins. *Protein Sci.*, 3:866–875, 1994.

[2] D. H. Andrews. The relation between the raman spectra and the structure of organic molecules. *Phys. Rev.*, 36:544–554, 1930.

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

[4] Aderonke Babajide, Ivo L. Hofacker, Manfred J. Sippl, and Peter F. Stadler. Neutral networks in protein space: A computational study based on knowledge-based potentials of mean force. *Folding & Design*, 2:261–269, 1997. Santa Fe Institute Preprint 96-12-085.

[5] Res. T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philos. Trans. R. Soc. Lodon*, 53:370–418, 1763.

[6] F.C. Bernstein, T.F. Koetzle, G.J.B. Williams, E.F.Jr Meyer, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112(3):535–542, 1977.

[7] M. Berrera, H. Molinari, and F. Fogolari. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4:8, 2003.

[8] S.D. Black and D.R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Anal. Biochem.*, 193:72–82, 1991.

[9] M. Born and R. Oppenheimer. Zur Quantumtheorie der Molekeln. *Ann. Phys. (Leipzig)*, 84:457–484, 1927.

[10] James U. Bowie, Neil D. Clarke, and Carl O. Pabo. Identification of protein folds: Matching hydrophobicity patterns of sequence sets with solvent accessibility patterns of known structures. *Proteins*, 7:257, 1990.

[11] James U. Bowie, Roland Lüthy, and David Eisenberg. Assessment of protein models with three-dimensional profiles. *Nature*, 356:83–85, 1992.

[12] J.U. Bowie, R. Luthy, and D. Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 235:164–70, 1991.

[13] C. Bradford, Barber, David P. Dobkin, and Hannu T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22:469–483, 1996.
`URL: http://www.acm.org`.

[14] B. R. Brooks, R. E. Bruccoleri, B. D Olafson, D. J. States, S. Swaminathan, and M Karplus. CHARMM: A program for macromolecular energy minimization and dynamics calculations. *J. Comp. Chem.*, 4:187–217, 1983.

[15] Joseph D. Bryngelson. When is a potential accurate enough for structure prediction? theory and application to a random heteropolymer model of protein folding. *J. Chem. Phys.*, 100:6038–6045, 1994.

[16] C.W.Jr Carter, B.C. LeFebvre, S.A. Cammer, A. Tropsha, and M.H. Edgell. Four-body potentials reveal protein-specific correlations to stability changes caused by hydrophobic core mutations. *JMB*, 311:625–638, 2001.

[17] Georg Casari and Manfred J. Sippl. Structure-derived hydrophobic potential: Hydrophobic potentials derived from X-ray structures of globular proteins is able to identify native folds. *J. Mol. Biol.*, 224:725–732, 1992.

[18] Hue Sun Chan and Ken A. Dill. The effects of internal constraints on the configurations of chain molecules. *J. Chem. Phys.*, 92,5:3118–3135, Mar. 1990.

[19] Hue Sun Chan and Ken A. Dill. Comparing folding codes for proteins and polymers. *Proteins*, 24:335–344, 1996.

[20] G. M. Crippen. Prediction of protein folding from amino acid sequence over discrete conformation spaces. *Biochemistry*, 30:4232–4237, 1991.

[21] Gordon M. Crippen and Yoshiaki Zenmei Ohkubo. Statistical mechanics of protein folding by exhaustive enumeration. *Proteins*, 32:425–437, 1998.

[22] I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annuals of Probability*, 3:146–158, 1975.

[23] Ken A. Dill, S. Bromberg, K. Yue, K. M. Fiebig, D. P. Yeo, P. D. Thomas, and H. S. Chan. Principles of protein folding: a perspective from simple exact models. *Prot. Sci.*, 4:561–602, 1995.

[24] I. Dostrovsky, E. D. Hughes, and C. K. Ingold. The role of steric hindrance. Section G. magnitude of steric effects, range of occurrence of steric and polar effects, and place of the wagner rearrangement in nucleophilic substitution and elimination. *J. Chem. Soc.*, page 173, 1946.

[25] S. E. Fienberg. An iterative procedure for estimation in contingency tables. *Annals of Math. Stat.*, 41:907–917, 1970.

[26] E. Furuichi and P. Koehl. Influence of protein structure database on the predictive power of statistical pair potentials. *Proteins: Struct. Funct. Genet.*, 31:139–149, 1998.

[27] Richard A. Goldstein, Zaida A. Luthey-Schulten, and Peter G. Wolynes. Optimal protein-folding codes from spin-glass theory. *Proc. Natl. Acad. Sci.*, 89:4918–4922, 1992.

[28] Richard A. Goldstein, Zaida A. Luthey-Schulten, and Peter G. Wolynes. Optimized energy functions for tertiary structure prediction and recognition. In H.Bohr and S.Brunak, editors, *Protein Structure by Distance Analysis*. IOS Press, 1994.

[29] S. Govindarajan, R. Recabarren, and R.A. Goldstein. Estimating the total number of protein folds. *Proteins: Struct. Funct. Genet.*, 35:408–414, 1999.

[30] T. Grossman, R. Farber, and A. Lapedes. Neural net representations of empirical protein potentials. *Ismb*, 3:154–61, 1995.

[31] K. Grünberger. *A 3D-Model for coarse grained Structure Prediction of RNA*. PhD thesis, University of Vienna, 2002.

[32] M. Hendlich, P. Lackner, S. Weitckus, H. Floeckner, R. Froschauer, K. Gottsbacher, G. Casari, and M. Sippl. Identification of native protein folds amongst a large number of incorrect models. *J. Mol. Biol.*, 216:167–180, 1990.

[33] T. L. Hill. Steric Effects. *J. Chem. Phys.*, 14:465, 1946.

[34] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of a representative set of structures from the brookhaven protein data bank. *Protein Science*, 1:409–417, 1992.

[35] I. L. Hofacker. personal communication, 1999.

[36] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.

[37] Martijn A. Huynen, Peter F. Stadler, and Walter Fontana. Smoothness within ruggedness: the role of neutrality in adaptation. *Proc. Natl. Acad. Sci.*, 93:397–401, 1996.

[38] C. T. Ireland and S. Kullback. Contingency tables with given marginals. *Biometrika*, 55:179–188, 1968.

[39] B. Krishnamoorthy and A. Tropsha. Development of a four-body statistical pseudo-potential to discriminate native from non-native protein conformations. *Bioinformatics*, 19(12):1540–1548, 2003.

[40] T. Lazaridis and M. Karplus. Discrimination of the native from misfolded protein models with an energy function including implicit solvation. *J. Mol. Biol.*, 288:477–487, 1998.

[41] S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18:534–552, 1985.

[42] Peter J. Munson and Raj K. Singh. Statistical significance of hierarchical multi-body potentials based on delaunay tessellation and their application in sequence-structure alignment. *Protein Science*, 6:1467–1481, 1997.

[43] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[44] B. Park and M. Levitt. Energy functions that discriminate x-ray and near native folds from well-constructed decoys. *J. Mol. Biol.*, 258:367–392, 1996.

[45] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comp. Phys. Commun.*, 91:1–41, 1995.

[46] J. W. Ponder and Case. D. A. Force fields for protein simulation. *Adv. Prot. Chem.*, 66, 2003. in press.

[47] Raj K. Singh, Alexander Tropsha, and I. I. Vaisman. Delaunay tessellation of proteins: Four body nearest neighbor propensity of amino acid residues. *J. Comp. Biol.*, 3:213–221, 1996.

[48] M. J. Sippl, Maria Ortner, Markus Jaritz, Peter Lackner, and Flöckner Hannes. Helmholz free energies of atom pair interactions in proteins. *Folding & Design*, 1:289–298, 1996.

[49] Manfred J. Sippl. Calculation of conformational ensembles from potentials of mean force — An approach to the knowledge-based prediction of local structures in globular proteins. *J. Mol. Biol.*, 213:859–883, 1990.

[50] Manfred J. Sippl. Boltzmann's principle, knowledge-based mean fields and protein folding. an approach to the computational determination of protein structures. *Journal of Computer-Aided Molecular Design*, 7:473–501, 1993.

[51] Manfred J. Sippl. Recognition of errors in three-dimensional structures of proteins. *Proteins*, 17:355–362, 1993.

[52] J.M. Smith. Natural selection an the concept of protein space. *Nature*, 225:563–564, 1970.

[53] G. Song, S. Thomas, K. A. Dill, J.M. Scholtz, and N. M. Amato. A path planning-based study of protein folding with a case study of hairpin formation in protein g and l. In *Pacific Symposium on Biocomputing*, volume 8, pages 240–251, 2003.

[54] P. Spelucci. A new technique for inconsistent qp problems in the sqp method. *Math. Method. Oper. Res.*, 47(3):355–400, 1998.

[55] P. Spelucci. An sqp method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82:413–448, 1998.

[56] W. A. Svrcek-Seiler. personal communication, 2003.

[57] S. Tanaka and H. A. Scheraga. Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. *Macromolecules*, 9:945–950, 1976.

[58] P. D. Thomas and K. A. Dill. Statistical potentials extracted from protein structures: How accurate are they? *J. Mol. Biol.*, 257:457–469, 1996.

[59] J. D. Van der Waals. *Over de Continuiteit van den Gas- en Vloeistoftoestand.* PhD thesis, Leiden, 1873.

[60] N. von Öhsen, I. Sommer, and R. Zimmer. Profile-profile alignment: A powerful tool for protein structure prediction. In *Pacific Symposium on Biocomputing*, volume 8, pages 252–263, 2003.

[61] G. Weberndorfer. Empirical protein potentials from delaunay tessellation. Master's thesis, University of Vienna, 1999.

[62] G. Weberndorfer, I. L. Hofacker, and P. F. Stadler. An efficient potential for protein sequence design. *In Giegerich et al (eds.), Computer Science in Biology. [GCB] '99 Proceedings*, pages 107–112, 1999.

[63] J. Xu, M. Li, G. Lin, D. Kim, and Y. Xu. Protein threading by linear programming. In *Pacific Symposium on Biocomputing*, volume 8, pages 264–275, 2003.

[64] C. T. Zhang. Relations of the number of protein sequences, families and folds. *Protein Eng*, 10:757–761, 1997.

[65] L. Zhang and J. Skolnick. What should the Z-score of native protein structures be? *Protein Science*, 7:1201–1207, 1998.

[66] W. Zheng, S. J. Cho, I. I. Vaisman, and Alexander Tropsha. Statistical geometry analysis of proteins: implications for inverted structure prediction. In Lawrence Hunter and Teri Klein, editors, *Biocomputing: Proceedings of the 1996 Pacific Symposium*, pages 614–23. World Scientific Publishing Co, 1996.

[67] W. Zheng, S. J. Cho, I. I. Vaisman, and Alexander Tropsha. A new approach to protein fold recognition based on delaunay tessellation of protein structure. In Lawrence Hunter and Teri Klein, editors, *Biocomputing: Proceedings of the 1997 Pacific Symposium*, pages 486–97. World Scientific Publishing Co, 1997.

[68] Ralf Zimmer, Marko Wöhler, and Ralf Thiele. New scoring schemes for protein fold recognition based on voronoi contacts. *Bioinformatics*, 14:295–308, 1998.

[69] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.

A Amino Acids

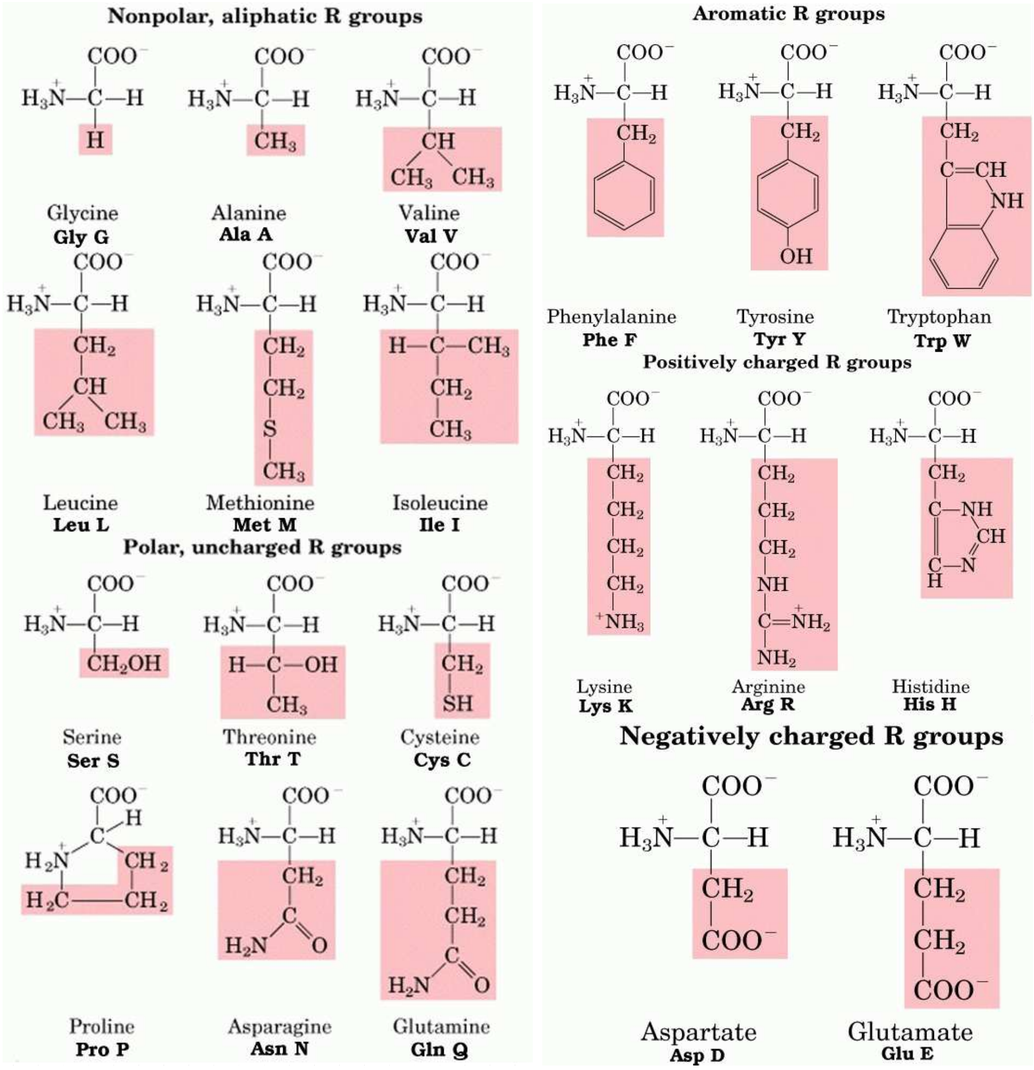


Figure 29: The twenty amino acids regularly building proteins. R-groups in the red boxes denote the side chains. Amino acids are arranged according to the functional groups of the side chain. Name as well as three letter and one letter code are beneath the respective amino acid.

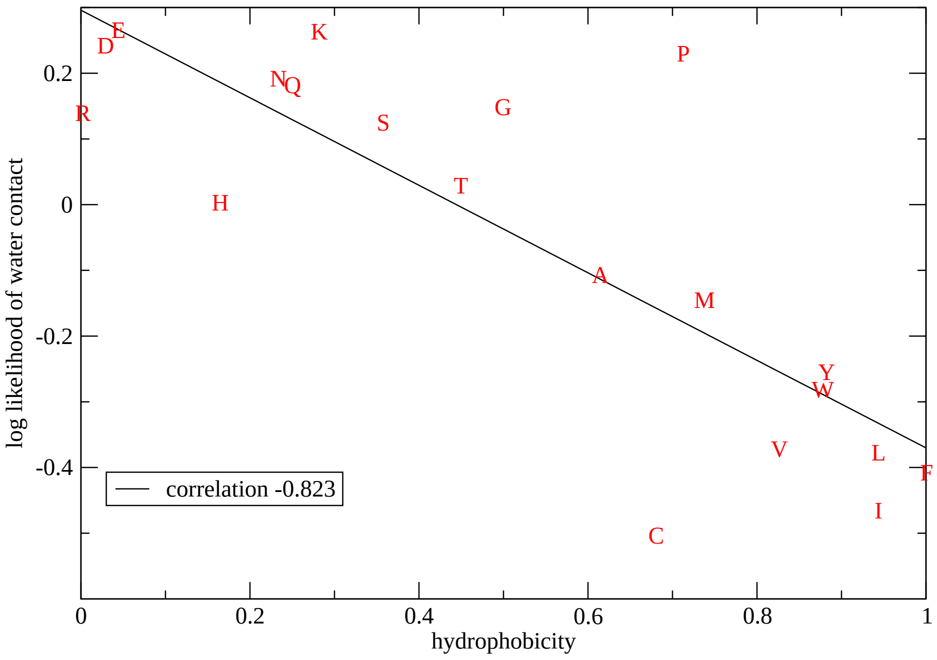


Figure 30: Hydrophobicity against water contact potential of amino acids. If **P** and **C** are left out, correlation is -0.922

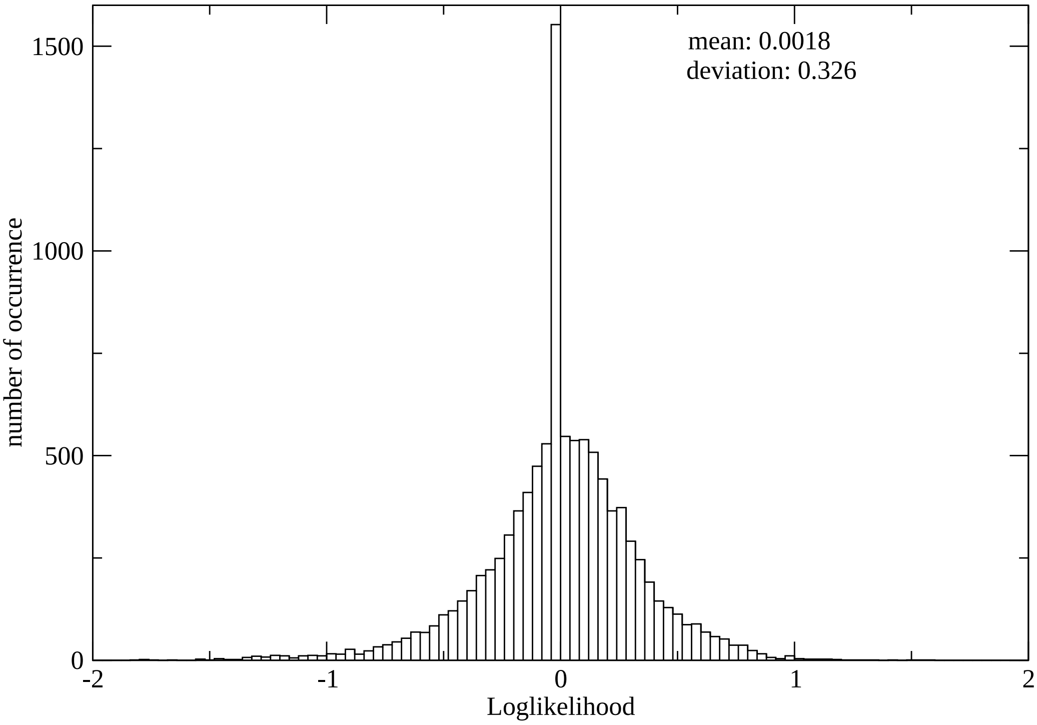


Figure 31: Distribution of the 10395 log-likelihoods of the tetrahedrons in the `SaDSaT` Potential. The big number of Log-likelihoods equal to zero (970) is due to sparse data correction applied.

	Ala	Cys	Asp	Glu	Phe	Gly	His	Ile	Lys	Leu	Met	Asn	Pro	Gln	Arg	Ser	Thr	Val	Trp	Tyr
Ala	0.254	-0.091	-0.156	-0.112	0.135	-0.056	-0.073	0.186	-0.156	0.198	0.126	-0.099	-0.191	-0.097	-0.066	-0.081	-0.034	0.173	0.038	0.021
Cys		1.763	-0.224	-0.324	0.239	0.0584	0.158	0.090	-0.189	0.045	-0.026	-0.041	0.118	-0.091	0.033	0.067	0.001	0.131	0.310	0.208
Asp			-0.244	-0.282	-0.066	-0.187	0.034	-0.148	-0.003	-0.185	-0.134	-0.067	-0.257	-0.201	-0.021	-0.128	-0.137	-0.142	-0.038	-0.039
Glu				-0.192	-0.133	-0.300	-0.124	-0.072	0.096	-0.059	-0.133	-0.232	-0.216	-0.188	0.059	-0.195	-0.169	-0.115	-0.030	-0.102
Phe					0.465	-0.007	0.190	0.391	-0.082	0.392	0.347	-0.008	0.011	0.006	0.029	0.013	0.062	0.374	0.286	0.378
Gly						-0.040	-0.080	-0.074	-0.205	-0.126	-0.136	-0.071	-0.091	-0.194	-0.135	-0.110	-0.023	-0.092	0.009	-0.055
His							0.431	0.062	-0.260	0.119	0.165	-0.177	-0.075	-0.045	-0.023	-0.008	0.046	0.028	0.202	0.134
Ile								0.636	-0.078	0.468	0.334	-0.057	-0.100	-0.024	-0.019	-0.052	0.131	0.514	0.150	0.289
Lys									-0.156	-0.041	-0.085	-0.160	-0.317	-0.168	-0.251	-0.254	-0.200	-0.094	-0.147	0.008
Leu										0.543	0.285	-0.139	-0.090	0.031	0.073	-0.079	0.074	0.399	0.186	0.230
Met											0.381	-0.165	-0.076	-0.052	0.062	-0.106	-0.064	0.283	0.275	0.191
Asn												0.052	-0.101	-0.124	-0.151	-0.097	-0.044	-0.139	0.086	-0.012
Pro													-0.000	-0.166	-0.161	-0.111	-0.086	-0.045	0.095	0.026
Gln														0.103	-0.056	-0.123	-0.025	-0.089	0.159	0.007
Arg															-0.022	-0.143	-0.097	-0.047	0.107	0.020
Ser																0.002	-0.066	-0.061	0.061	0.049
Thr																	0.069	0.107	0.054	0.028
Val																		0.442	0.249	0.226
Trp																			0.564	0.345
Tyr																				0.312

Amino Acid	Hydrophobicity ([8])	Loglikelihood of water contact
Ala	0.616	-0.108
Cys	0.680	-0.505
Asp	0.028	0.243
Glu	0.043	0.262
Phe	1.00	-0.410
Gly	0.501	0.150
His	0.165	0.005
Ile	0.943	-0.465
Lys	0.283	0.266
Leu	0.943	-0.375
Met	0.738	-0.143
Asn	0.236	0.189
Pro	0.711	0.230
Gln	0.251	0.181
Arg	0.000	0.141
Ser	0.359	0.126
Thr	0.450	0.029
Val	0.825	-0.375
Trp	0.878	-0.277
Tyr	0.880	-0.260

Table 11: The Hydrophobicity [8] and the Log-likelihood of a water-amino acid contact

B Binomial Coefficients

β is needed because for `SaDSaT` there is no difference between, say, a triangle a, b, c and a triangle a, c, b, so all possible permutations of a, b and c are taken to be the same triangle. The number of possible permutations is given by β. List of binomial coefficients β used in this work:

$$\beta_{a,b} = \begin{cases} 1, & a = b \\ 2, & a \neq b \end{cases}$$

$$\beta_{a,b,c} = \begin{cases} 1, & a = b = c \\ 3, & a \neq b = c \\ 6, & a \neq b \neq c \end{cases}$$

$$\beta_{a,b,c,d} = \begin{cases} 1, & a = b = c = d \\ 4, & a \neq b = c = d \\ 6, & a = b \neq c = d \\ 12, & a \neq b \neq c = d \\ 24, & a \neq b \neq c \neq d \end{cases}$$

C Library

1 Introductions

SaDSaT calculates the z-score of an amino acid sequence on a 3dimensional structure, given by the coordinates of its $C\beta$ atoms (or virtual $C\beta$s in case of Glycine). It tessellates the structure as well as a the structures obtained by threading the sequence through a poly-protein together with a shell of virtual water molecules to simulate the surface of the protein. All the amino acid pairs existent in the tessellation are grouped in sequence and euclidean distance classes. The quadruples and pairs are then used to compute an energy or z-score using log-likelihood parameters SaDSaT acquires out of a data base. As well as this energy evaluation, tessellation, a statistical part of the program and a optimization part for obtaining expected frequencies of quadruples given the pair frequencies, inverse folding by means of adaptive walks is possible.

The program is built of 4 modules performing different tasks: SaDSaTstat is used to obtain statistical information out of data bases in PDB file format, CompLogLike uses this statistical information to compute log likelihood parameters, SaDSaT computes energies and z-scores and SaDSaTinv is used to run adaptive walks For those who wish to develop their own programs we provide a library which can be linked to your own code.
The stand-alone programs is described in a separate man page. This manual documents version 0.1.0.

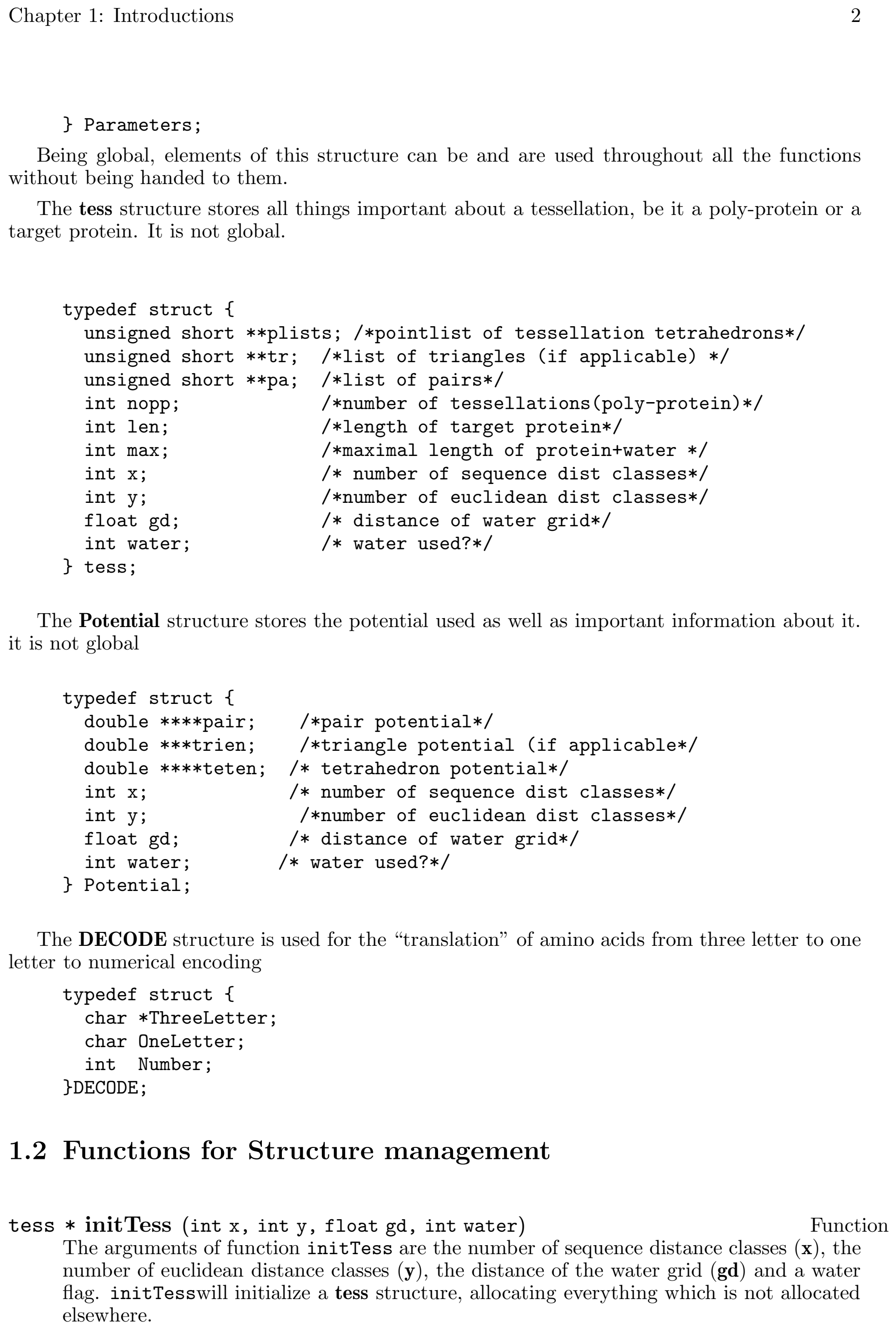
==

1.1 Structures used in the library

The library uses 4 different structures

Structure **Parameters** stores all the parameters and flags of the program, it is quite big, which may look clumsy, but, on the other hand, you always know where to find flags. This structure is global.

```
typedef struct{
  float gd;        /*distance of waters in grid*/
  int x;         /*number of sequence classes*/
  int y;         /*number of euclidean classes*/
  char *atom_type; /*target atom type*/
  char *ppname;    /*name of polyprotein file*/
  char wt_seq[MAXSEQLEN]; /*wild type sequence */
  int * seq_nc;    /*numerically encoded sequence*/
  int verbose;     /*output verbose or no??*/
  int water;     /*watershell?? no=0;*/
  char *tpp_file;   /*filename of tessellation to save */
  double watscale;  /*weight of water i.r.t. core */
  char *target_name; /*name of targed pdb file*/
  int seqcomp;      /* prompt for new sequence after computation?*/
  int vmd;         /*vmd_output yes/no*/
  char *potfn;     /*namepart of potential to load*/
  int SS;          /*use eisenberg's environment for mutation?*/
  int surf;        /*use surface/core for mutation*/
  int seq;        /*prompt for sequence adaptive walk is started from?*/
  int z;          /*compute z-score before adaptive walk?*/
  int restr;      /*prompt for mutation after computation*/
  char *outfile;  /*name of output file*/
  int energy;     /*compute energy only*/
  int tri;        /*use triangles?*/
```



```
} Parameters;
```

Being global, elements of this structure can be and are used throughout all the functions without being handed to them.

The **tess** structure stores all things important about a tessellation, be it a poly-protein or a target protein. It is not global.

```
typedef struct {
  unsigned short **plists; /*pointlist of tessellation tetrahedrons*/
  unsigned short **tr;  /*list of triangles (if applicable) */
  unsigned short **pa;  /*list of pairs*/
  int nopp;             /*number of tessellations(poly-protein)*/
  int len;              /*length of target protein*/
  int max;              /*maximal length of protein+water */
  int x;                /* number of sequence dist classes*/
  int y;                /*number of euclidean dist classes*/
  float gd;             /* distance of water grid*/
  int water;            /* water used?*/
} tess;
```

The **Potential** structure stores the potential used as well as important information about it. it is not global

```
typedef struct {
  double ****pair;    /*pair potential*/
  double ***trien;    /*triangle potential (if applicable*/
  double ****teten;  /* tetrahedron potential*/
  int x;             /* number of sequence dist classes*/
  int y;              /*number of euclidean dist classes*/
  float gd;          /* distance of water grid*/
  int water;        /* water used?*/
} Potential;
```

The **DECODE** structure is used for the “translation” of amino acids from three letter to one letter to numerical encoding

```
typedef struct {
  char *ThreeLetter;
  char OneLetter;
  int  Number;
}DECODE;
```

1.2 Functions for Structure management

`tess *` **initTess** (`int x, int y, float gd, int water`) Function

The arguments of function `initTess` are the number of sequence distance classes (**x**), the number of euclidean distance classes (**y**), the distance of the water grid (**gd**) and a water flag. `initTess`will initialize a **tess** structure, allocating everything which is not allocated elsewhere.

`Potential *` **get_potentials** (`int seqcl, int eukcl, float gd, int water, char *fname`) Function

The arguments of function `get_potentials` are the number of sequence distance classes (**seqcl**), the number of euclidean distance classes (**eukcl**), the distance of the water grid (**gd**), a water flag and the changeable part of the name of the potential files (**fname**), usually **Parameters.potfn**. `get_potentials` will read the potentials from disk, scanning the current working directory for the following files, depending on **Parameters.tri**: if tri=no (default): **pair**[**fname**][**seqcl**][**eukcl**] and **scndtet**[**fname**] if tri=yes: **pair**[**fname**][**seqcl**][**eukcl**], **tri**[**fname**] and **tet**[**fname**] if one of these does not exist, the function will cause the program to exit. If the parameters (**seqcl**, **eukcl** and **gd**) given do not match the parameters in the file, a warning will be printed and the parameters will be changed to the ones in the file. Note that this may cause seqfolds later.

`void` **freeTess** (`tess tess`) Function

The argument is a **tess** structure, which this function frees.

`void` **freePot** (**Potential** `pot`) Function

The argument is a **Potential** structure, which this function frees.

1.3 Input/Output Functions

`void` **make_vmdoutput** (`unsigned short *pl, float *coords, int len`) Function

The arguments of `make_vmdoutput` are the pointlist to print (one of the **tess.plists**, the coordinates of the $C\beta$s, read by `read_pdb` and the length of the target protein. `make_vmdoutput` will create 2 files, [**targetname**]**.vmd** containing the edges of the tessellation, and [**targetname**]**.vmd2**, containing the nodes. [targetname] is the basename of **par.target_name**. Both files will use different colors for surface and core.

`int` **save_PP_tessellation** (`char *fname, tess *tmp, int len, int ctrl`) Function

The arguments of `save_PP_tesselation` are the name of the saved tessellation file to create, which will be appended by .tpp and is stored in **par.tpp_file** usually, the tessellation to be saved, the length of the target protein and a control integer to tell `save_PP_tessellation` whether it should try to save triangles. `save_PP_tessellation` will write the tessellation in a binary file readable by `read_PP_tessellation`, it will not only save pointlist and pairlists (and trianglelists), but also crucial information about how the tessellation was computed, so that it cannot be used unknowingly for computations with different parameters.

`tess *` **read_PP_tesselation** (`char *fname, tess *tmp, int len`) Function

The arguments of `read_PP_tessellation` are the name of the file to be read, a pointer to a **tess** function initialized using `initTess` and the length of the target protein. The function will check whether the parameters `save_PP_tessellation` wrote in the file are the same the user is using, and will terminate the program if this is not the case.

`float *` **read_pdb** (`char *pdb_name, char *seq`) Function

`read_pdb` is the PDB-parser. The arguments it needs are the name of the PDB-file to read as well as a pointer to the sequence. If the sequence pointer is a NULL pointer, it will not read the sequence. It returns the coordinates of the $C\beta$ or $C\alpha$ atoms of the protein in the PDB-file, creating virtual $C\beta$ for glycine. The array of coordinates has the sequence length as its first entry **cord**[**0**], the next three **cord**[**1**], **cord**[**2**], **cord**[**3**] are the coordinates of the first atom and so on. If a char ***seq** pointer is given, the sequence will be saved in one capital letter code. `read_pdb` tries to detect inconsistencies in PDB-files (and there are at least some). It will check the distance between two neighboring $C\alpha$ or $C\beta$ atoms to find gaps not obviously denoted by gaps in sequence numbers. It will always use the "A" possibility if applicable, i.e. it will not read in the second of two possible coordinate sets for an amino acid. If you want to use the "B" coordinates, edit the PDB-file. When finding gaps, `read_pdb` will return an array with -1 as first entry, thus denoting that reading the file failed. It will save selected reasons in an error message. Non standard amino acids can not be read. It will parse only up to the first TER signal.

`void` **get_new_sequence** (`int len, char *sequence`) Function

`get_new_sequence` prompts for a new sequence in CAPITAL one letter code, and translates it to numerical encoded sequence in the **wt_seq** global array. The arguments are the length of the target sequence and the sequence character pointer.

1.4 Tessellation related Functions

`tess *` **tessellate** (`float *coords, int len, tess *temp`) Function

`tessellate` performs the tessellation of a target or poly-protein. The arguments are the coordinates in the format created by `read_pdb`, the length of the target protein and a **tess** structure. This function will perform the tessellation, creating the pointlists and writing them in **tess.plists**. It calls different other functions we also provide.

`float *` **make_watershell** (`float *coords, float gd`) Function

`make_watershell` creates a shell of water surrounding the protein. The arguments are the coordinates as created by `read_pdb` and the distance of the water in the grid in Angstrom. It will return a new, longer set of coordinates, including the coordinates of the water shell. This function is used in `tessellate`.

`unsigned short *` **call_qhull** (`int numpoints, coordT array[][4]`) Function

`call_qhull` is the function which calls qhull, the algorithm performing the tessellation. The arguments are the number of points to be tessellated and an two dimensional array of these coordinates, i.e. **atom[0][0]** is the x coordinate of the first atom, **atom[0][1]** the y and so forth. Note that coordT has to be set in the header to match the definition in the qhull library. We use double as coordT. It will return a list of four point contacts, with the first ([0]) element 4 times the number of tetrahedrons it encountered, and the four-tuples (e.g. [1][2][3][4]) the nodes participating in a tetrahedron. Also used in `tessellate`.

`unsigned short *` **processpointlist** (`unsigned short *pl, int watst`); Function

`processpointlist` will truncate the pointlist, leaving no "water only" tetrahedrons. The arguments are the old pointlist and the length of the target protein (the point water starts). This function is used in `tessellate`.

`tess *` **gettrli** (`tess *temp`) Function

`gettrli` will make a list of unique triangles out of a pointlist of tetrahedrons. The argument is a pointer to a **tess** structure with an existing **tess.plists** (out of e.g. tessellate). The triangle list **tess.tr** is an array of arrays (for each tessellation one in case of a poly-protein, else only one) with the length of the array as **[x][0]** entry and then tri-tuples describing the triangles (i.e. **[x][1]**,**[x][2]** and **[x][3]** are one triangle).

tess * **getpali** (`tess *temp, float *coord`) Function

`getpali` will make a list of unique pairs out of a pointlist of tetrahedrons. The arguments are a pointer to a **tess** structure with an existing **tess.plists** (out of e.g. tessellate) and the coordinates of the points of the pointlist(`read_pdb` created). Water coordinates are not needed.

The pairlist **tess.pa** is an array of arrays (for each tessellation one in case of a poly-protein, else only one) with the length of the array as **[x][0]** entry and then tri-tuples describing the pairs (i.e. **[x][1]**,**[x][2]** the nodes of the pair and **[x][3]** the euclidean class).

1.5 Energy related Functions

double **dtdznew** (`tess *PP, tess *targ, Potential *pot`) Function

This function returns the z-score of the **tess** targ protein on the **tess** PP poly-protein with the **Potential** pot. It needs the global structure **wt_seq**. If the flag **Parameters.verbose** is set, the function will put out splitted z-scores and energies also.

double * **initialize_zscore** (`tess *PP, Potential *pot, int *seq`) Function

`initialize_zscore` will return the mean energy and the mean energy squared of the tessellation of a poly-protein **tess** PP. The arguments are said **tess** PP, the Potential **Potential** pot and the numerical encoded sequence.

The mean energy is returned on [0], the mean energy squared on [1].

double * **compute_energy** (`tess *temp, Potential *pot, int* seq`) Function

`compute_energy` computes the energy of the sequence int seq on the tessellation **tess** temp with the potential **Potential** pot. It will return an array of energies corresponding to the number of different tessellations put in (i.e. for a poly-protein it will return on [0] the first, on [1] the second energy and so forth). For a single tessellation (i.e. the target protein), the energy is put in [0].

This function is used in `initialize_zscore`.

double **zscohr** (`double Targen, double *Mw`) Function

`zscohr` computes a z-score out of an energy **Targen** and an array containing the mean energy and the mean energy squared (as computed in `initialize_zscore`).

1.6 Functions for adaptive walks

int **mutateseq** (`char *seq, int *no`) Function

`mutateseq` will mutate an amino acid of sequence seq. It will use the frequency of natural occurrence to determine which amino acid will randomly be chosen to replace a randomly chosen amino acid. The integer array no contains all amino acids which may be mutated. If all amino acids may be mutated, it has its length on first position and then subsequent numbers: e.g. [15][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15] if e.g. amino acids 3 4 5 should not be mutated, it will look like this: [12][1][2][5][6][7][8][9][10][11][12][13][14][15].

int **mutateseqS** (`char *seq, int *no, char *Struc`); Function

`mutateseqS` will mutate an amino acid of sequence seq. It will use the frequency of natural occurrence dependent on the position regarding surface or core to determine which amino acid will randomly be chosen to replace a randomly chosen amino acid. The integer array no contains all amino acids which may be mutated. The character string **Struc** is a string of **S** and **C** for surface and core, as long as the amino acid sequence.

`int` **mutateseqE** (`char *seq, int *no, char *Struc`); Function

`mutateseqE` will mutate an amino acid of sequence seq. It will use the frequency of natural occurrence dependent on the environmental class to determine which amino acid will randomly be chosen to replace a randomly chosen amino acid. The integer array no contains all amino acids which may be mutated. The character string Struc is a string where each amino acid of a sequence has 3 characters, describing the environmental class, as long as the amino acid sequence.

1.7 Other useful functions

`int` **get_top_par** (`int a, int b, int parx`); Function

This returns the sequence distant class of two amino acids. The arguments are the number of the amino acids **a** and **b** as well as the number of sequence distance classes to use. The boundaries of this classes can be edited if need be.

`int` **get_distanceclass** Function

(double x1, double y1, double z1, double x2, double y2, double z2, int pary) This returns the euclidean distance class two amino acids belong to. The arguments are the coordinates of the two amino acids and the number of euclidean distance classes. Boundaries can be edited.

`void` **dostat** (`tess tess, char *outfile, int *seq`) Function

This function is used to do statistics. It will read in files with the names aa[**outfile**][**par.x**][**par.y**], pair[**outfile**][**par.x**][**par.y**], tri[**outfile**] and tet[**outfile**] and will add the information contained in **tess** tess. The last argument is the numerically encoded sequence. Default out-name is cnt. If these files do not exist, it will create them.

`int *` **addwatertoseq_nc** (`int *seq_nc, int aamolnumber, int seq_length`) Function

`addwatertoseq_nc` will take a numerically encoded sequence seq_nc and add water from sequence position seq_length+1 to sequence position aamolnumber.

1.7.1 Global Variables and structures

These global variables may be needed by any of the functions above. It is strongly recommended to use them.

`int` **trop_err_code** Variable

An integer to save an error code.

`char *` **program** Variable

the name of the program

`DECODE *` **Rosetta** Variable

For translation.

`char` **qh_version**[] = "**3.1 <02/05/01>**"; Variable

The version information for qhull

Parameters par holds parameters

`char` **trop_err_msg[MAX_ERR_MSG_LEN]** Variable
Will hold an definition of the error, on the off-chance that one occurs *g*

`char *` **alphabet /* points to Amino acids or AA-classes */** Variable
the amino acid alphabet used (Usually 20 one letter code: ACDEFGHIKLMNPQRSTVWY), initialize right after start of program.

`int` **filter_delauney** Variable
Will tessellation use filtering??

`int` **wt_seq[MAXSEQLEN]** Variable
holds numerically encoded sequence.

`int *` **globseq** Variable
holds numerically encoded sequence.

D Programs

D.1 `SaDSaTstat`

NAME

SaDSaTstat − reads a pdbfile, performs a tessellation and saves the number of amino acids, pairs, triangles and tetrahedrons it encountered, either in a new file or after reading old files in and adding the numbers up.

SYNOPSIS

SaDSaTstat [options] pdb−file

DESCRIPTION

The **SaDSaTstat** program reads the coordinates and sequence of a protein in a PDB−file, then performs a tessellation, counts all occurrences of amino acids, amino acid pairs, −triangles and−tetrahedrons. Then it either reads in existing files and adds the new numbers to them, or creates a new file and saves this numbers. The files created can be used by **CompLogLike** to generate a log likelyhood potential. Additional information such as the number of sequence− and euclidean distance classes as well as whether water was used and wat the distance of the water grid was are saved also to prevent usage of wrong potentials. The program always uses one pdb−file. It is recomended to use a shell script if whole data bases have to be processed.

a small foreach loop can make usage of **SaDSaTstat** easy:

```
foreach i ('*.pdb')
SaDSaTstat [options] $i
end
```

will do a statistics over all pdb−files in a directory

OPTIONS

−−TA [CA|CB]

Specify targed atom, either CA or CB are possible. Note that the parameters?? should be created with the same targed atom.

−−pot [name]

The name of the files to load and/or write. Default is cnt. So the files aa[name][SDC][EDC], pair[name][SDC][EDC], tri[name] and tet[name] will tried to be read, and if not existing, created.

−−verbose −v

Print more information.

−−SDC [Number]

Specify the number of sequence distance classes to be used in the computation. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same number of sequence distance classes.

−−EDC [Number]

Specify the number of euclidean distance classes to be used in the computation. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same number of euclidean distance classes.

−−GD [Number]

Specify the distance of the water molecules in the shell. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same distance.

−−water −w

Do not compute watershell and hence do not compute a surface part of the potential.

FILES

Files created and/or read by **SaDSaTstat**

```
aa[name][SDC][EDC]
pair[name][SDC][EDC]
tri[name]
tet[name]
```

the files tri[inname] and tet[inname] will be of the following form:

```
[GD] # griddistance
0 # triangle  0– 0– 0
1 # triangle  0– 0– 1
0 # triangle  0– 0– 2
1 # triangle  0– 0– 3
and so on
[GD] #griddist
0 # tet  0– 0– 0– 0
0 # tet  0– 0– 0– 1
0 # tet  0– 0– 0– 2
and so on
```

With first the grid distance saved and then all found tetrahedrons/triangles

pair[inname][SDC][EDC] and aa[inname][SDC][EDC] start like this:

```
[SDC] #seqdistcl
[EDC] #eukdistcl
[GD] #griddistance
0 #  aa 0 seq 0 euk 0
3 #  aa 1 seq 0 euk 0
and so on
[SDC] #seqdistcl
[EDC] #eukdistcl
[GD] #griddistance
1 # pair  0/ 0 seq  0 euk  0
4 # pair  0/ 1 seq  0 euk  0
and so on
```

The program will try to open the files specified by −−pot to add the counted numbers to the numbers saved there. If the [SDC] and [EDC] are not equal to the values used, it will crash. If the files do not exist, it will make them.

REFERENCES

The tessellation is done using qhull algorithm (http://www.thesa.com/software/qhull/)

A. Tropsha had the idea to use tessellation for knowledge based potentials.

Much of this program has been influenced by M. Sippl's work.

P. F. Stadler and I. L. Hofacker are the geniuses whos ideas I used mainly.

Guenther Weberndorfer did much basic work.

C. Bradford, D. P. Barber and H. T. Huhdanpaa "The Quickhull Algorithm for Convex Hulls" ACM Transactions on Mathematical Software, 22, pp 469–483, 1996
R. K. Singh, A. Tropsha and I. I. Vaisman " Delauney Tessellation of Proteins: Four Body Nearest Neighbor Propensity of Amino Acid Residues" Journal of Computational Biology, 3, pp 213–221, 1996
M. J. Sippl "Calculation of Conformational Ensembles from Potentials of Mean Force – An Approach to the knowledge–based Prediction of Local Structures in Globular Proteins" Journal of Molecular Biology,

213, 859–883, 1990

VERSION

This man page documents Version 0.1.0 of SaDSaTstat

AUTHORS

Stephan Bernhart, Ivo L. Hofacker, Peter F. Stadler

BUGS

If in doubt and overstrung, try to yell and scream and run. Else, you can contact berni@tbi.univie.ac.at

SEE ALSO

SaDSaT, SaDSaTinv, CompLogLike

D.2 `CompLogLike`

NAME

CompLogLike – creates a potential file for **SaDSaT** with input created by **SaDSaTstat**

SYNOPSIS

CompLogLike [options]

DESCRIPTION

The **Comploglike** program reads the number of amino acids, amino acid pairs, amino acid triangles and amino acid tetrahedrons in a data base out of files created by **SaDSaTstat** for use with **SaDSaT(inv)**. It uses an optimizer, **donlp2** to compute expected frequencies for tetrahedrons, creating the energy contributions for pairs and tetrahedrons by comparing them with the observed frequencies after correcting for sparse data. The logarithm of the quotient of them is used as energy contribution. Some data of how the data was achieved is also saved to prevent mistakes in computations due to using wrong energy parameters for computing z–scores. The files created have names of the form pair[name][SDC][EDC] and scndtet[name].

OPTIONS

––water –w

The files to be read were not created using watershell, an @code{Comploglike} will therefor not try to computed energies for water contacts.

––verbose –v

Print more information.

––SDC [Number]

Specify the number of sequence distance classes to be used in the computation. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same number of sequence distance classes.

––EDC [Number]

Specify the number of euclidean distance classes to be used in the computation. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same number of euclidean distance classes.

––GD [Number]

Specify the distance of the water molecules in the shell. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same distance.

––potin [name]

The midle part of the name of the files created by @command{SaDSaTstat}.

––potout [name]

The midle part of the name of the files created by Comploglike to be used by SaDSaT and SaDSaTinv.

––help –h

Print an informative help message describing the options and then exit successfully.

FILES

files that have to be created by **SaDSaTstat** for **CompLogLike** to work:

aa[name][SDC][EDC]
pair[name][SDC][EDC]
tri[name]
tet[name]

the files tri[inname] and tet[inname] will be of the following form:

```
[GD] # griddistance
0 # triangle  0– 0– 0
1 # triangle  0– 0– 1
0 # triangle  0– 0– 2
1 # triangle  0– 0– 3
and so on
[GD] #griddist
0 # tet  0– 0– 0– 0
0 # tet  0– 0– 0– 1
0 # tet  0– 0– 0– 2
and so on
```

With first the grid distance saved and then all found tetrahedrons/triangles

pair[inname][SDC][EDC] and aa[inname][SDC][EDC] start like this:

```
[SDC] #seqdistcl
[EDC] #eukdistcl
[GD] #griddistance
0 #  aa 0 seq 0 euk 0
3 #  aa 1 seq 0 euk 0
and so on
[SDC] #seqdistcl
[EDC] #eukdistcl
[GD] #griddistance
1 # pair  0/ 0 seq  0 euk  0
4 # pair  0/ 1 seq  0 euk  0
and so on
```

files created by **CompLogLike**:

pair[outname][SDC][EDC]
scndtet[outname]

scndtet[outname] takes the same form as tet[inname]:

```
[GD] #
–0.283478 #  0 –  0 –  0 –  0
and so forth
```

while pair[outname][SDC][EDC] is analogous to pair[inname][SDC][EDC]:

```
[SDC] # SDC
[EDC] # EDC
[GD] # GD
0.589936 # 0,0,0,0
and so forth
```

REFERENCES

A. Tropsha had the idea to use tessellation for knowledge based potentials.

Much of this program has been influenced by M. Sippl's work.

P. F. Stadler and I. L. Hofacker are the geniuses whos ideas I used mainly.

Guenther Weberndorfer did much basic work.

donlp2 is available at ftp://ftp.mathematik.tu–darmstadt.de/pub/department/software/opti/

R. K. Singh, A. Tropsha and I. I. Vaisman " Delauney Tessellation of Proteins: Four Body Nearest Neighbor Propensity of Amino Acid Residues" Journal of Computational Biology, 3, pp 213–221, 1996

M. J. Sippl "Calculation of Conformational Ensembles from Potentials of Mean Force − An Approach to the knowledge−based Prediction of Local Structures in Globular Proteins" Journal of Molecular Biology, 213, 859−883, 1990
P. Spelucci "An sqp method for general nonlinear programs using only equality constrained subproblems" Mathematical Programming, 82, pp 413−448, 1998
P. Spelucci "A new technique for inconsistent qp problems in the sqp method" Math. Method. Oper. Res., 47, pp 355−400, 1998

AUTHORS

Stephan Bernhart, Ivo L. Hofacker, Peter F. Stadler

BUGS

If in doubt and overstrung, try to yell and scream and run. Else, you can contact berni@tbi.univie.ac.at

SEE ALSO

SaDSaT, SaDSaTinv, SaDSaTstat

D.3 SaDSaT

NAME

SaDSaT – compute z–score or energy of an amino acid sequence on a 3–Dimensional structure.

SYNOPSIS

SaDSaT [options] pdb–file [poly–protein]

DESCRIPTION

SaDSaT program reads the coordinates and sequence of a protein in a PDB–file, then computes its energy or z–score using poly–protein files also in PDB format using energy parameters produced by the .I CompLogLike program. It prints the z–score or energy to standard out. Additional information such as the contributions of surface and core of the protein to the z–score can be read via options. Additional output such as a vmd–source file for visualization can be generated via options. In Addition, new sequences or mutations can be computed after computation of the original z–score/energy. The tessellation of the poly–protein can be saved.

OPTIONS

––TA [CA|CB]

Specify targed atom, either CA or CB are possible. Note that the parameters?? should be created with the same targed atom.

––savePP [Filename]

Save the tessellation of the poly–protein to disk. This greatly reduces the time neede for computation, but needs quite a lot of disk space (˜200M for the 10k poly protein). The file will get the extension .tpp. If the poly–protein given in the command line has an .tpp extension, a previously saved tessellation will be read. It is important not to try and use a saved tessellation which was computed for a protein of a different length and/or different SDC, EDC or GDs (see below)

––water –w

Do not compute watershell and hence do not compute a surface part of the potential.

––verbose –v

Print more information.

––SDC [Number]

Specify the number of sequence distance classes to be used in the computation. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same number of sequence distance classes.

––EDC [Number]

Specify the number of euclidean distance classes to be used in the computation. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same number of euclidean distance classes.

––GD [Number]

Specify the distance of the water molecules in the shell. Note that the parameters?? as well as a save poly–protein tessellation must be created with the same distance.

––outvmd –o

Creates vmd source files for visualization of the tessellation. A file name.vmd containing the edges and a file name.vmd2 containing the nodes of the tetrahedrons are created, with the edges and nodes in different colours if they belong to the water shell. Type source name.vmd(2) in the vmd console to view it.

−−help −h
Print an informative help message describing the options and then exit successfully.

−−potin [name]
Type in the middle part of the potential files (created by **CompLogLike**), they have the form pair[name][SDC][EDC] and scndtet[name]. Default name is log. If they are computed without watershell, a ’w’ is appended to this names.

−−watscale [factor]
Determine the scale of water compared to surface term. This factor will always be multiplied by GDˆ4/3. The default is 0.05.

−−energy −e
Do not compute z−score, but an energy of the protein only. Because there is no threading procedure, there is no need of a poly−protein file name. Because the verbose mode will cause the programm to crash, it is automatically turned off if −e is chosen.

−−triangle −t
Use triangles in computation. If this is chosen, other potential files will be read in, namely tri[name] and tet[name] instead of scndtet[name] are needed!

−−seq −s
This is used to compute the energy of another sequence at the structure. After finishing computation (either energy or z−score), the program will prompt for the input of another sequence for which the computation is then repeated, and another sequence is prompted until @ as terminating character is inserted. The sequence has to be inserted in one letter CAPITALS. Wrong characters will cause the program to crash terminally.

−−res −r
This is used to introduce one point mutations into the sequence and rerun computation. As with −s after finishing computation (either energy or z−score), the program will prompt for input. The input has two parts: first, the number of the amino acid to be mutated is to be inserted, then, what amino acid (in CAPITAL one letter code) is to replace the native one. The output is the energy difference together with what kond of mutation has been made (in numerical mode).

FILES

The files needed by **SaDSaT**: Potential files created by **CompLogLike**, named pair[name] and scndtet[name] (else see **−t**), a target file in pdb−format and a poly−protein file either in pdb or in **.tpp** format (as created by the **−−savePP** option. The files have to have been created with the same **[SDC]**, **[EDC]** and **GD** as well as, in case of a **.tpp** file, with the same length of the target protein.

REFERENCES

The tessellation is done using qhull algorithm (http://www.thesa.com/software/qhull/)

A. Tropsha had the idea to use tessellation for knowledge based potentials.

Much of this program has been influenced by M. Sippl’s work.

P. F. Stadler and I. L. Hofacker are the geniuses whos ideas I used mainly.

Guenther Weberndorfer did much basic work.

C. Bradford, D. P. Barber and H. T. Huhdanpaa "The Quickhull Algorithm for Convex Hulls" ACM

Transactions on Mathematical Software, 22, pp 469−483, 1996
R. K. Singh, A. Tropsha and I. I. Vaisman " Delauney Tessellation of Proteins: Four Body Nearest Neighbor Propensity of Amino Acid Residues" Journal of Computational Biology, 3, pp 213−221, 1996
M. J. Sippl "Calculation of Conformational Ensembles from Potentials of Mean Force − An Approach to the knowledge−based Prediction of Local Structures in Globular Proteins" Journal of Molecular Biology, 213, 859−883, 1990

VERSION

This man page documents Version 0.1.0 of SaDSaT

AUTHORS

Stephan Bernhart, Ivo L. Hofacker, Peter F. Stadler

BUGS

If in doubt and overstrung, try to yell and scream and run. Else, you can contact berni@tbi.univie.ac.at

SEE ALSO

SaDSaTinv, SaDSaTstat, CompLogLike

D.4 `SaDSaTinv`

NAME

SaDSaTinv
– computes an adaptive walk to find a sequence which will fold into a given structure.

SYNOPSIS

SaDSaTinv [options] pdb−file poly−protein

DESCRIPTION

The **SaDSaTinv** program reads the coordinates and sequence of a protein from a PDB−file, then computes as well as from a poly−protein files also in PDB format. Using energy parameters produced by the **CompLogLike** program, it then begins with a random sequence to search for a sequence fitting this structure by adaptive walk, i.e. randomly mutating single amino acids and keeping the mutation if the z−score of the new sequence is better than that of the old one. It will stop if it cannot find a mutation increasing the z−score in 100 tries. The output is a file containing the sequences and zscores of the adaptive walk.

OPTIONS

−−TA [CA|CB]

Specify targed atom, either CA or CB are possible. Note that the parameters?? should be created with the same targed atom.

−−savePP [Filename]

Save the tessellation of the poly−protein to disk. This greatly reduces the time neede for computation, but needs quite a lot of disk space (˜200M for the 10k poly protein). The file will get the extension .tpp.

−−water −w

Do not compute watershell and hence do not compute a surface part of the potential.

−−verbose −v

Print more information.

−−SDC [Number]

Specify the number of sequence distance classes to be used in the computation. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same number of sequence distance classes. Default is 5

−−EDC [Number]

Specify the number of euclidean distance classes to be used in the computation. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same number of euclidean distance classes. Default is 2

−−GD [Number]

Specify the distance of the water molecules in the shell. Note that the parameters?? as well as a save poly−protein tessellation must be created with the same distance. Default is 5.1 (Angstrom)

−−Sequence

The program will prompt for a starting sequence to be started from. It has to be put in in one letter code. Capital letters will be mutated, lower case characters will not. This can be useful if you want to design a protein with certain features at a certain region.

−−Surface

When mutating, use different probabilitys for surface anc core amino acids. The sequence of surface and core amino acids has to be put in after prompted. −−Surface and −−Secstruc (see below) are mutually exclusive.

−−Secstruc

When mutating, use different probabilitys for different environmental classes (see Eisenberg). The sequence of environmental classes has to be put in after prompted. −−Surface (see above) and −−Secstruc are mutually exclusive.

−−zscore −z

Compute z−score of the wild type sequence (or the sequence contained in the trarget pdb−file) first.

−−help −h

Print an informative help message describing the options and then exit successfully.

−−potin [name]

Type in the middle part of the potential files (created by **CompLogLike**), they have the form pair[name][SDC][EDC] and scndtet[name]. Default name is log. If they are computed without watershell, a 'w' is appended to this names.

−−outfile [name]

The name of the output file, will be appended with .out. Default is adaout.

−−watscale [factor]

Determine the scale of water compared to surface term. This factor will always be multiplied by GDˆ4/3. The default is 0.05.

−−triangle −t

Use triangles in computation. If this is chosen, other potential files will be read in, namely tri[name] and tet[name] instead of scndtet[name] are needed!

FILES

The files needed by **SaDSaTinv**: Potential files created by **CompLogLike**, named pair[name] and scndtet[name] (else see **−t**), a target file in pdb−format and a poly−protein file either in pdb or in **.tpp** format (as created by the **−−savePP** option. The files have to have been created with the same **[SDC]**, **[EDC]** and **GD** as well as, in case of a **.tpp** file, with the same length of the target protein.

The output file will contain:

[Consecutive Number] [energy] sdc [SDC] edc [EDC] gd [GD] [−t]
[SEQUENCE] (in one letter code)
[z−score]

REFERENCES

The tessellation is done using qhull algorithm (http://www.thesa.com/software/qhull/)

A. Tropsha had the idea to use tessellation for knowledge based potentials.

Much of this program has been influenced by M. Sippl's work.

P. F. Stadler and I. L. Hofacker are the geniuses whos ideas I used mainly.

Guenther Weberndorfer did much basic work.

C. Bradford, D. P. Barber and H. T. Huhdanpaa "The Quickhull Algorithm for Convex Hulls" ACM Transactions on Mathematical Software, 22, pp 469−483, 1996
R. K. Singh, A. Tropsha and I. I. Vaisman " Delauney Tessellation of Proteins: Four Body Nearest Neighbor Propensity of Amino Acid Residues" Journal of Computational Biology, 3, pp 213−221, 1996
M. J. Sippl "Calculation of Conformational Ensembles from Potentials of Mean Force − An Approach to the knowledge−based Prediction of Local Structures in Globular Proteins" Journal of Molecular Biology, 213, 859−883, 1990

VERSION

This man page documents Version 0.1.0 of SaDSaTinv

AUTHORS

Stephan Bernhart, Ivo L. Hofacker, Peter F. Stadler

BUGS

If in doubt and overstrung, try to yell and scream and run. Else, you can contact berni@tbi.univie.ac.at

SEE ALSO

SaDSaT, SaDSaTstat, CompLogLike

List of Figures

1 Four component picture of inter- and intra-molecular forces . . 6
2 Eisenberg's environmental classes 12
3 Tessellations in 2 dimensions 22
4 A Voronoi Diagram and the corresponding Delaunay tessellation 25
5 Delaunay tessellation of 1ab1 26
6 Water excluding procedure 40
7 Delaunay tessellation of a square grid 41
8 Water-shell of 1ab1 42
9 Water molecules added per amino acid 43
10 Z-scores of data base proteins 46
11 Water molecules added as a function of grid distance 51
12 Water pairs against g 52
13 Z-scores against grid distance 53
14 Distribution of euclidean distance 54
15 Number of pairs against sequence separation 55
16 Z-score as a function of detail 56
17 RMSD z-score correlation 61
18 Correlation between experimental $\Delta\Delta G$ and computed energy difference 64
19 Adaptive walks 10k 30k 66
20 Adaptive walks 10k 66
21 Adaptive walks 30k 67
22 Adaptive walks comparing Weberndorfer's to SaDSaT potential 68
23 Comparison of adaptive walks using `PROSAII` and `SaDSaT` . . . 70
24 rmsd/delta E 74
25 RMSD z-score correlation 75
26 rmsd/delta E 76
27 RMSD z-score correlation 77
28 Comparison of adaptive walks using `SaDSaT` 78
29 The 20 amino acids i
30 Hydrophobicity against water contact potential of amino acids ii

31 Distribution of tetrahedral log-likelihoods iii

List of Tables

1 Eisenberg's scoring matrix 13
2 SCOP classes of PDB Select proteins 47
3 Boundaries of euclidean distance classes 54
4 Boundaries of sequence distance classes 55
5 Z-scores of different proteins using different potentials 57
6 Comparison of z-score of proteins with length 129 59
7 Performance of different potentials on decoy sets 59
8 Performance on decoy sets for different `SaDSaTs` 62
9 Performance of different potentials on decoy sets 73
10 Log likelihoods of amino acid pairs iv
11 Hydrophobicity and water contact potential of amino acids . . v

Curriculum Vitae

Full Name:	Stephan Helmut Franz Bernhart
Date and place of birth:	12.03.1976 in Vienna, Austria
Education:	Volksschule Herzgasse Wien 10 Bundesgymnasium Ettenreichgasse Wien 10
Matura:	June 1994, passing with distinction
Military service	Oct.1994 - Apr.1995 Austrian army
Studies:	1995 - 2001 Biochemistry, University of Vienna 2001 - 2003 Chemistry, University of Vienna
Address:	Institut für Theoretische Chemie Währingerstr 17 A - 1090 Wien berni@tbi.univie.ac.at