Protein Folding

1 Introduction

Proteins are made of a linear chain of amino acids, but their 3-D structure is almost never linear or a random coil. The vast majority of proteins fold into its characteristic 3-D shape. The well-defined 3-D shape or fold is determined by the interactions among the amino acids in the protein. The 3-D fold of the protein, known as its *native state*, is believed to be the thermodynamically stable state. The native structure is essential for the correct function of the protein. Without it, the protein will not function correctly and It was shown, in a well-known experiment by Anfinsen, that the amino-acid sequence of a protein determines its native conformation and that most proteins will fold into the same 3-D native structure under the same physiological conditions. A *conformation* refers to any possible 3-D arrangement of the protein.

Anfinsen's Experiment and the Thermodynamic Hypothesis

In the 1960's, Christian B. Anfinsen and his coworkers at the National Institutes of Health (NIH) performed a series of seminal experiments in vitro that answered a key part of the problem. In these experiments, they denaturated ribonuclease A using urea and 2-mercaptoethanol (2ME). These are reducing agents that caused the protein to unfold by breaking disulfide bridges. They showed that the enzyme would re-assume structure and enzymatic activity after denaturation, once the reducing agents were removed [Anfinsen(1973)]. Many other proteins have since been shown to be able to be renatured in similar experiments.

The original work led Anfinsen to propose his *"Thermodynamic Hypothesis"*, which states that the "biologically-active conformation" or native conformation of a protein, is adopted spontaneously under a "normal physiological milieu" which includes the solvent, pH, ionic strength, presence of other components such as metal ions or prosthetic groups, temperature, etc. In other words, there is sufficient information contained in the protein sequence to guarantee correct folding from any of a large number of unfolded states. In 1972, Anfinsen received the Nobel prize in chemistry for his formulation of this relationship between the amino acid sequence and the biologically-active (functional) structure of a protein.

Anfinsen's hypothesis states that the native conformation of a protein is the one in which the *Gibbs free energy* is the lowest. Another way to think about it is: As a protein folds, the free energy goes down – but what is free energy?

The (Gibbs) free energy is defined as G = E - TS

- E = potential energy (interatomic interactions)
- T = temperature (on which folding happens)
- S = entropy (measures degeneracy of a state)

2 Folding Pathways and Landscapes

Several hypotheses tried to explain the way proteins fold from a denatured, random conformation into their native state. The classical view of protein folding describes the folding process as a

pathway or a series of discrete intermediates which follow in sequential order to the native structure. According to this model, the information on which pathways the protein takes is encoded somehow in the unfolded states.

An alternative model, called the energy landscape theory of folding, considers folding as the progressive organization of an ensemble of partially folded structures through which the protein passes on its way to the natively folded structure. The energy landscape shows that the range of states becomes more limited with lower energy, and the landscape view encompasses the preferred pathways.

If proteins search for conformations, they do so on an energy landscape similar to a multidimensional, rugged funnel-like structure. The funnel view of protein folding was simultaneously proposed by Ken Dill and Peter Wolynes [Wolynes(1997), Dill and Chan(1997)]. According to the model, the slope of the funnel guides the protein down towards the energy minimum, where the native state resides. The folding funnel leads to increase in rates of protein folding compared to expected rates for random diffusion processes The folding funnel also largely prevents entrapment of partially folded states (local energy minima), implied by the ruggedness of the folding funnel.

The landscape model of protein folding was confirmed by experiments and is now well accepted. This is the model that drives the many computer algorithms that fold proteins *in silico*.

According to this model, as more sections of the native structure fall into place, the protein achieves lower energy, with the final folded native structure at the bottom of the funnel. The landscape view allows thinking of multiple molecules rolling down the pathways of successive local minima until they all converge to the global minimum. This process closely resembles the experiment, where measurements are obtained over multiple protein molecules (replicas). The width of the funnel (cross-section) gives the entropy – a measurement of how many different conformations achieve similar energy value.

Some experiments suggest that folding does not always lead to a unique state or structure corresponding to the overall free-energy minimum. The *kinetic hypothesis* suggests that a protein may get trapped in a local minimum in a really rugged energy landscape. Some proteins are assisted in their folding by a class of proteins called *chaperones* (other proteins that oversee and correct the folding process)

2.1 How Does a Protein Chain Fold?

Since proteins do not seem to fold at random, there must be an order of events that drives the folding of a protein chain. An important question arises: what drives the folding of a protein chain?

Several models exist here too: The **framework model** states that local interactions drive the folding process. There is an order of events: Secondary structures form first and then, once formed, they stay that way. The stable secondary structures then self-assemble in the native folded conformation. An alternative theory, the **collapse model**, Proposed by Ken Dill in 1985 [Dill and Chan(1997)], states that non-local interactions drive the folding process. In other words – global collapse drives the secondary structure formation and not the other way around. According to this theory, the "folding code" that drives the sequence to its native structure, resides mainly in global patterns of contact interactions that are non-local.

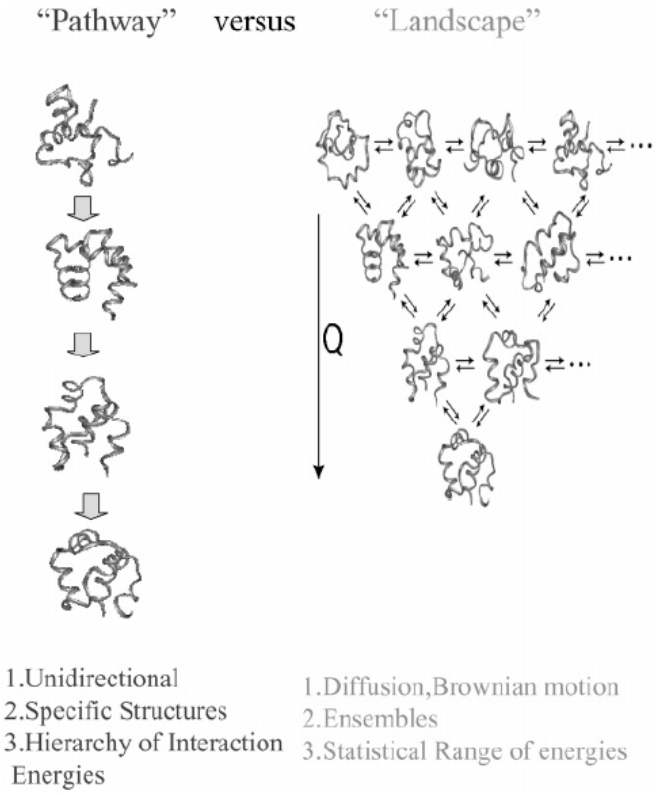


Figure 1: Pathway vs. Landscape folding theories

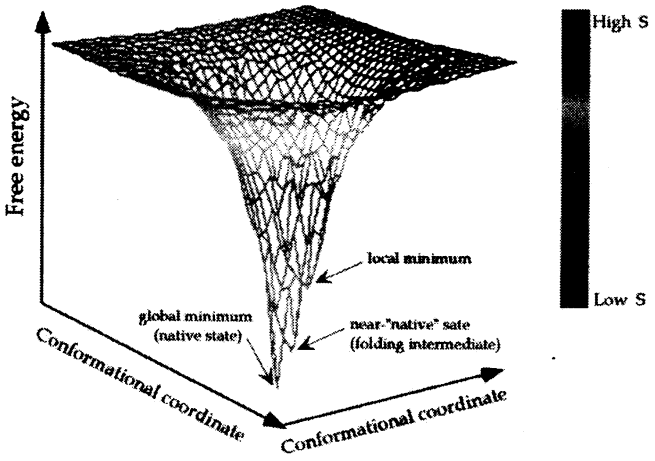


Figure 2: The funnel view of protein conformational landscape

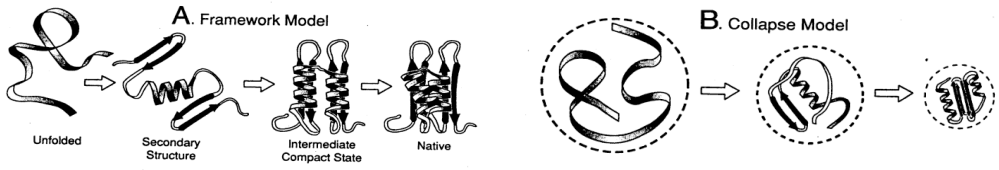


Figure 3: The funnel view of protein conformational landscape

2.2 Case Study: What Happens When it All Goes Wrong?

The eligibility criteria for blood donation from the red cross website contain the following paragraph:

"You are not eligible to donate if:

From January 1, 1980, through December 31, 1996, you spent (visited or lived) a cumulative time of 3 months or more, in the United Kingdom (UK), or From January 1, 1980, to present, you had a blood transfusion in any country(ies) in the (UK) or France."

Why is that? The answer is what is commonly known as "The Mad Cow" disease. A bovine epidemic struck the UK in 1986. 170,000 cows appeared to be "mad": they drooled and staggered, were extremely nervous, or bizarrely aggressive. Eventually, they all died. As the brains of the dead "mad" cows was analyized, the researchers realized they resembled a sponge. The disease was called bovine spongiform encephalopathy, or BSE. Several years prior, In 1982 the infectious agents responsible for transmitting spongiform encephalopathy were defined and named prions (Stanley Prusiner, 1982).

Other examples of spongiform encephalopathy are scrapie which develops in sheep, Creutzfeld-Jacob Disease (CJD) and its variant (vCJD) which develop in humans and causes similar symptoms, resulting in death.

Prions are proteins that are found in the nerve cells of all mammals. Many abnormally-shaped prions are found in the brains of BSE-infected cows and vCJD or CJD patients. The difference in normal and infectious prions may lie in the way they fold. Evidence indicates that the infectious agent in transmissible spongiform encephalopathy is a misfolded protein – a protein that folds into an incorrect 3D shape. The normal protein is called PrPC (for cellular). Its secondary structure composition is dominated by alpha helices.

The abnormal, disease producing protein called PrPSc (for scrapie), has the same primary structure (amino acid sequence) as the normal protein, but its secondary structure is dominated by beta conformations. How can this happen?

The abnormally-shaped prion gets absorbed into the bloodstream and crosses into the nervous system. The abnormal prion touches a normal prion and changes the normal prion's shape into an abnormal one, thereby destroying the normal prion's original function. Both abnormal prions then contact and change the shapes of other normal prions in the nerve cell. The nerve cell then tries to get rid of the abnormal prions by clumping them together in small sacs. Because the nerve cells cannot digest the abnormal prions, they accumulate in the sacs that grow and engorge the nerve cell, which eventually dies. When the cell dies, the abnormal prions are released to infect

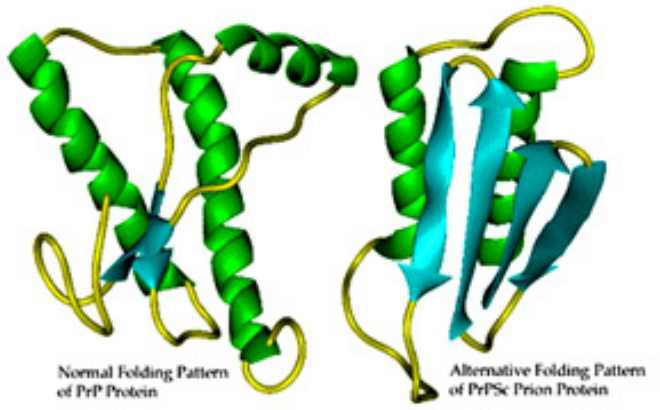


Figure 4: A normal (left) and affected (right) prion.

other cells. Large, sponge-like holes form where many cells die. In other words, an affected prion "infects" a normal prion.

The Prion Hypothesis suggests that diseases like mad cow and human CJD are caused by the misfolding of a protein known as PrP that most cells contain. Once a few copies of the protein become misfolded), they cause other PrPs to misfold, leading to an accumulation of insoluble proteins in the cell. misfolded proteins cause cell death and damage the nervous system.

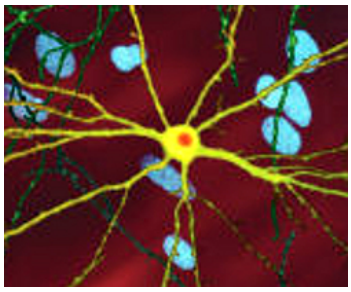
Not Only Cows: Human Prion Diseases

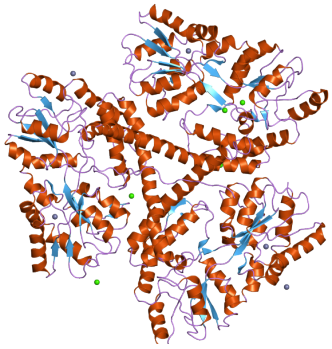
Creutzfeldt-Jakob disease (CJD) is a rare fatal brain disorder that usually occurs in late life and runs a rapid course. There is no known treatment or cure. Most CJD cases are sporadic (not hereditary), but about 5 to 10% of cases are due to an inherited genetic mutation associated with CJD (familial CJD).
The mutation makes the prion protein more susceptible to misfolding. How and why? Still not entirely clear... It can also be acquired through contact with infected brain tissue (iatrogenic CJD) or consuming infected beef, hence the blood donation restrictions.

Similar yet Different: Huntington's Disease

Huntington's disease is a rare and (usually) hereditary disease that causes death of brain cells It has an autosomal dominant inheritance pattern: One copy of the mutated gene is enough to cause the disease. The symptoms include: Mood disorders, uncoordinated movements, and eventually dementia and death. It is a late-onset disease. The typical age at onset is 30–50 but it can appear earlier or later. The life expectancy is 15-20 years from diagnosis

The gene responsible for Huntington's disease (HTT) contains a sequence of CAG – repeated multiple times (i.e. ... CAGCAGCAG ...) This sequence codes to the amino acid Glutamine (GLN, Q). This creates a sequence of Glutamines, called PolyQ (polyglutamine chain). The number of repeats varies normally in the population. An abnormally large number of CAG repeats in the HTT gene causes Huntington's disease.





Repeat count	Classification	Disease status	Risk to offspring
<26	Normal	None	None
27–35	Intermediate	None	Elevated but << 50%
36–39	Reduced Penetrance	Maybe	50%
40+	Full Penetrance	Will be affected	50%

A sequence of 36 or more glutamines results in the production of a protein which has different characteristics. The PolyQ regions appears to adopt a β-sheet structure This altered form, called mutant huntingtin (mHTT), increases the decay rate of certain types of neurons. The huntingtin protein interacts with over 100 other proteins, and appears to have multiple biological functions. Enzymes in the cell often cut the elongated protein into fragments, which form abnormal clumps inside nerve cells, and may attract other, normal proteins into the clumps (Sounds familiar?)

3 Computational Modeling of Protein Folding

The biggest question for computer science is – if a protein can spontaneously find its lowest energy conformation, how can we model the process and predict the native conformation of a given protein? Modeling how proteins fold into their native structures, known as the **protein folding problem**, has challenged computer scientists for decades. While progress has been made, the problem is still largely unsolved, especially when trying to fold a protein *ab initio*, based on physico-chemical information only, without prior knowledge about the native structure.

The main challenges for computer scientists, when trying to model the folding process, can be divided into roughly three overall categories: Structural representation, Evaluating the conformations, and search algorithms.

3.1 Structural Representation

The first thing we have to keep in mind is how to computationally represent the protein structure How are protein conformations represented computationally? The way protein structures are represented in the Protein Data Bank is using a set of 3D coordinates defining the geometric location of all the atoms in the protein (often with the exception of hydrogen atoms), but we can decide to model the protein structure using only a subset of the atoms rather than all of them to save time.

Why would we do that? The answer is simple. If all the atoms are explicitly modeled and followed in space, this puts a lot of computational burden on a search algorithm. In many cases it

is worth to sacrifice some of the accuracy for the sake of efficiency. Coarse-grained modeling (using only some atoms) vs. all-atom (explicit) modeling is an important decision.

However, if not all the atoms are explicitly modeled, how does one determine which ones to model? There are many possible coarse-grained models of protein structures: For example: Model every amino acid by a single sphere, centered at the C-α atom. Alternatively, model the backbone explicitly and represent the side chain as a single sphere. Many other coarse grained models are available.

Another question is – are the modeled atoms allowed to move anywhere in space? For Example: We can force atoms to be on a lattice – this brings enumeration into the realm of feasible computation, at least for small proteins. This made lattice-based models very popular in the earlier days. Lattice based models will be discussed in detail below. Most modern day structure prediction and folding algorithms use off-lattice models, which are more realistic and allow the atoms to move more freely in space, not confined to a lattice.

Coarse grained models allow us to quickly rule out impossible conformations, and we can then focus on the rest of the conformational space in more detail.

3.2 Ranking the Conformations

Remember the thermodynamic hypothesis, which states that a protein folds into the structure which has the lowest energy. How do we model the energy function on a computer? We have to come up with a *scoring function* to determine whether a conformation is of low or high energy. Such a function must fulfill the following requirements:

1. It has to be an actual mathematical function from the conformation space to the real numbers. In other words, it should be able to accept any protein conformation and output a numerical score.

2. It must do so efficiently (as we will see shortly, we may potentially evaluate a very large number of possible conformations!).

3. Last but certainly not least – it has to provide a good approximation of the physical energy of the protein.

Practical scoring functions are empirical AND there are many of them. In what follows we will use energy functions and scoring functions interchangeably, but you should keep in mind that most of these scoring functions do not accurately represent the thermodynamic quantity that is the Gibbs free energy. Rather, they try to approximate or estimate it. The scoring function you select depends a lot on the way you choose to represent the protein.

For efficiency purposes, it is very important to design functions that can accurately score coarse-grained conformations. Evaluation of an energy function on an all-atom representation is quite expensive. Several examples of energy functions will be given below. The subject is also further discussed later on.

3.3 Sampling Algorithm

One of the most important aspects in computationally folding a protein is the computational algorithm itself. The goal of a protein folding or structural prediction algorithm is to try to compute the native structure of a protein molecule based on its amino acid sequence. This is almost always

done by generating a very large number of possible low energy structures following some sampling scheme and evaluating them. If the goal is to also model the folding process, the algorithm should generate a set of successive structures that come closer and closer to the possible minimum energy conformation.

Degrees of Freedom (DOFs) How do we generate multiple conformations of a protein, given its amino acid sequence? Before we start explaining how to perform the sampling, we should lay down some ground definitions. When modeling protein folding computationally, the process generally involves changing or moving around some parts of the protein structure which we believe are free to move about.

Definition 1 (Degrees of Freedom) *The degree of freedom (DOF) is the set of independent parameters that can be varied to define the state of the system*

For example:

- The location of a point in a 2-D cartesian system has two independent parameters – its (x, y) coordinates. If these two coordinates are given, we can fully specify the location of the point (Figure 5 (a)). Similarly, the location of a point in a 3-D cartesian system has three independent parameters – (x, y, z).
- Polar coordinates are an alternative way to represent points using two independent parameters. We describe the point as lying on a circle with radius r, centered at the origin. θ is the angle the point makes with respect to the x axis, see Figure 5 (b) and (c). It is possible to switch from polar to cartesian coordinates using the following equations:

$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$

To switch from cartesian to polar coordinates, use the following equation:

$$r = \sqrt{(x^2 + y^2)}$$
$$\theta = tan^{-1}(y/x)$$

A molecule with n atoms can be represented by a set of $3 \times n$ cartesian coordinates, so it has $3 \times n$ DOFs... ...or does it? As a matter of fact, the atoms are mutually restricted by bond lengths and angles, and each such bond/angle poses a constraint on the system. For example – if we know that the bond length (distance) between two atoms is 1.5Å , then they are no longer independent! Therefore, the "real" number of DOFs is much smaller than $3 \times n$. To simulate folding, we have to manipulate the "real" degrees of freedom of the protein, usually a subset of the backbone dihedral angles. This brings the number of degrees of freedom down to $2 \times N - 2$, where N is the number of amino acids. As we shall see, it is still a lot.

Enumeration and Cyrus Levinthal's Paradox: Cyrus Levinthal suggested a famous thought experiment in 1969, noting that proteins have an enormous number of possible conformations, while only one (or very few) native conformation(s). Let us try to compute the native conformation of a small, 100 amino acid protein, by systematically sampling its conformational space. Assume

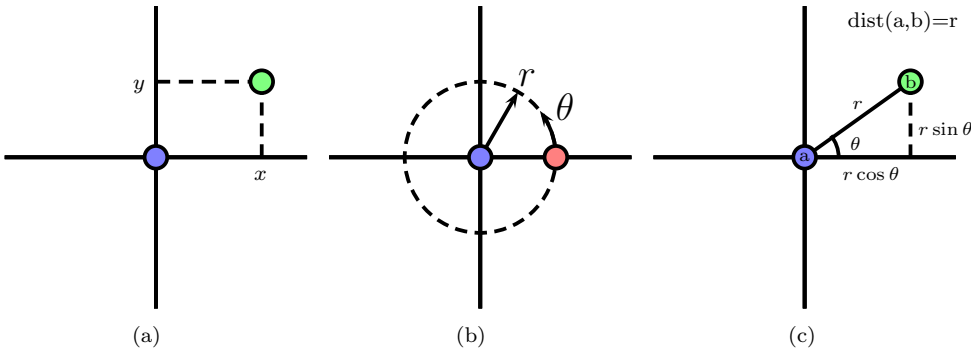


Figure 5: Examples of degrees of freedom: a. A point in a 2-D cartesian coordinate

for simplicity that we only consider the two backbone dihedral degrees of freedom as rotatable. Therefore, the protein has 2*100-2 = 198 torsional degrees of freedom. Suppose we wish to sample the entire space spanned by these 198 degrees of freedom in order to find the minimum energy conformation that corresponds to the native structure. Let us make another approximation and discretize the 360° space into units of 120°, which gives us only three possible dihedral positions per rotatable bond. Hence, even with this very coarse grained modeling, in order to sample the entire conformational space of this protein, we would need to sample 3^{198} different conformations! Even if conformations are sampled very rapidly, say a picosecond (10^{-12} of a second) per conformation, a full sampling would still require more time than the age of the universe! This is indeed impossible, as we know that proteins fold spontaneously over the course of several milliseconds (10^{-3} of a second) or less. How can it be?

Or is it really a Paradox? The only way out of the paradox is to realize that a protein that folds spontaneously does not systematically sample its entire conformational space on its way to its native state. Even Levinthal himself, being aware of the thermodynamic hypothesis, suggested that: "protein folding is sped up and guided by the rapid formation of local interactions which then determine the further folding of the peptide; this suggests local amino acid sequences which form stable interactions and serve as nucleation points in the folding process." [Levinthal(1968)]

In other words, there is method to the madness. There must be an energetic bias that "guides" the protein towards the biologically-active conformation without going through unnecessary conformations What does the energy landscape look like? What does this mean for how proteins actually fold?

Protein folding intermediates and partially folded transition states were experimentally detected, which explains the fast protein folding. Several theories have since been suggested, trying to explain the way proteins fold quickly into their native structures. These theories will be discussed below. Based on the folding theories, computational models to protein folding and structure prediction tried to identify and simulate the mechanism of protein folding, and predict the native conformation of proteins.

Structure Prediction vs. Folding: At this point, we should probably pause to make one small but important distinction between folding and structure prediction. The two terms are often being used interchangeably, but they are not exactly the same.

Protein folding refers to the process of folding, which leads to the native structure.
Structure prediction refers to the end result only.
In this chapter we will discuss methods for both folding and structure prediction.

3.4 Sampling Algorithms

The rest of the chapter will be dedicated to structure prediction algorithms. Many such algorithms exist, and they differ not only in the way they represent the protein molecule and in their energy model, but also in the amount of prior information they assume about the protein structure. Protein structure prediction algorithms vary from homology modeling methods, which use similar proteins with a known structure as a template to fold a sequence with an unknown structure, all the way to *ab-initio* methods, which try to fold a protein based on physical principles only. As you may guess, the more prior information you have, the more accurate and reliable your results will be. However, your output will be limited by the availability of such prior information.

3.4.1 Trajectory Based Sampling

Many sampling algorithms launch trajectories from a starting conformation, trying to get to the bottom of the folding basin by optimizing the energy function. Most trajectory-based explorations follow this general scheme:

1. Start with a random conformation
2. Generate the next one using a search algorithm
3. Rank the new conformation using the energy/scoring function, decide whether this conformation is acceptable
4. Continue until a convergence criterion is satisfied or for a specified number of iterations

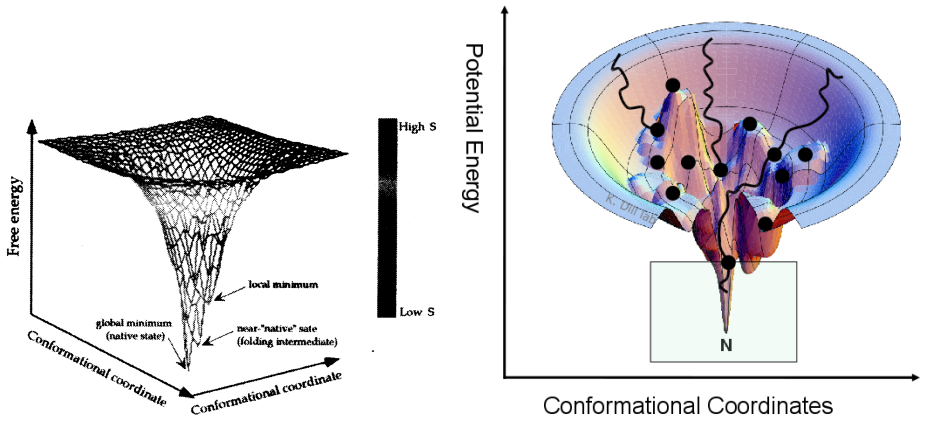
The search algorithm, and often the starting conformation, determine the performance and the outcome of our search. See Figure 6

The most commonly used search algorithms are either a systematic search using Molecular Dynamics (MD), or a probabilistic walk – using a Monte Carlo (MC) search. In MD, consecutive conformations in the trajectory are generated by following the gradient of the energy function. In MC, moves are sampled from a move set to generate the next conformation in the trajectory. The generated conformation is then accepted according to the Metropolis criterion (see below).

3.4.2 Systematic Search: Molecular Dynamics (MD)

Bio-molecular simulations will be discussed in detail later. Here we give a short description of the main stages:

Classical MD simulations use Newton's laws of motion to derive the change in position, velocity and acceleration of each atom in the molecule. It employs thermodynamics and statistical mechanics to simulate the folding process. The energy function used by MD simulation is usually empirical, physics based. Detailed, all atom simulations follow the motion of each atom in space including the solvent, even though there are coarse grained MD algorithms.



(a) (b)

Figure 6: (a) An illustration of the funnel model. (b) Launching multiple trajectories in search of the native structure.

Due to their physics-based and detailed nature, MD simulations reveal detailed information about the folding process but their usefulness in simulation protein folding or exploring the conformational space is limited. The conformational space is high-dimensional: there are many atoms to follow in space. An average sized protein contains several thousands atoms, and if we add the solvent molecule, we can easily reach 50,000 atoms or more. Therefore, MD simulations demand a lot of computer time to simulate even several nanoseconds of the folding process, whereas most proteins take at least several milliseconds to fold.

Locating the Basin of the Funnel with MD:

1. evaluate forces on each atom
2. move atom according to force numerical update: $x(t + dt) = x(t) + dt * f(x(t)) + ...$
3. go back to 1.

- Numerical update is the bottleneck
- dt needs to be small for accuracy (1-2 fs)
- Generates a single trajectory
- Results depend on initial conditions, since MD is essentially a local optimization technique.

3.4.3 Random (Probabilistic) Search: Monte Carlo (MC)

Monte Carlo is a well known computational method that relies on repeated random sampling. It can be used for optimization, in this case of the scoring function. It conducts biased probabilistic walks in the search space. Starting form some random conformation, at each point in time (iteration), a move is made that is not necessarily physical or representative of what the protein does to transition

between conformations. The move is a new conformation generated based on the current one. A conformation resulting from a move can either be accepted or rejected based on the optimization criteria. A popular optimization function is called the *Metropolis criteria*: If the energy of the new conformation is smaller (better) than the current one, accept the new conformation. Otherwise, accept it with a small probability, usually exponentially scaled by the energy difference. Due to the probabilistic nature of the method, the paths taken to the native state may be actually impossible for a protein to follow. However, By computing conformations through moves, MC exhibit higher sampling efficiency – it can make big jumps in conformational space, that are not possible for physics based method. Here is a description of the algorithm:

Metropolis Monte-Carlo (MMC)

1. Start with a random/extended conformation $C_{curr} \leftarrow C_{start}$
2. Make a move (change C_{curr}) which results in a new conformation C_{new}
3. Give a value to one of the parameters considered example: rotate a bond
4. If $E(C_{new}) < E(C_{curr})$ then $C_{curr} \leftarrow C_{new}$
5. else $dE = E(Cnew) - E(Ccurr)$
   $C_{curr} \leftarrow C_{new}$ with prob. $e^{-dE/scaling_factor}$
6. Goto 2.

Pros and Cons of MD and MC: Both MD and MC launch trajectories in conformational space The end-point of these trajectories depend to an extent on the initial conditions (conformations from which the trajectories were initiated) Both MD and MC are local optimization techniques aimed at sampling the global minimum in the energy landscape While MD gives physical trajectories, MC's trajectories may not correspond to a sequence of moves that a protein actually follows There may be nothing physical about the moves/parameters chosen to generate consecutive conformations This gives MC the ability to make bigger jumps in conformational space and sample the space faster than MD.

MD over MC? Try Enhanced Sampling

The basic exploration in MD and MC can be enhanced to avoid settling in local minima with enhanced sampling methods such as:

- Simulated annealing
- importance and umbrella sampling
- replica exchange (parallel tempering)
- local elevation
- activation relaxation
- local energy flattening
- jump walking

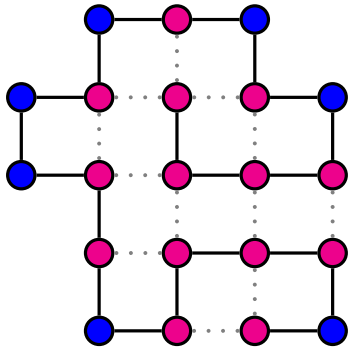


Figure 7: A 2-D lattice conformation. The blue beads represent polar amino acids and the magenta beads represent hydrophobic amino acids. dashed white lines represent H-H contacts

- multicanonical ensemble
- conformational flooding
- discrete timestep MD
- fragment-based assembly
- ... (many many more)

Oftentimes protein structure prediction and protein folding are used interchangeably to describe the prediction of a protein's native structure. However, these two terms are distinct from one another. Protein structure prediction is the prediction of the three-dimensional structure of a protein from its amino acid sequence. Protein folding refers to predicting the process in which a protein folds and not just the final, native structure.

3.5 Lattice Based Models – The Simplest Way to Fold

Our first folding example is simple. Since the conformational space is so vast, let us start with a simple model. Let's restrict our positions in space for an atom to a lattice or a grid. This simple model allows to enumerate all the possible conformations (on a short chain). The model can actually help answer the following two questions:

1. Which sequences lead to a low-energy native conformation (protein design)
2. Given a sequence, what is its native conformation?

HP Lattice Model: Ranking Sequences and Folds The HP (Hydrophobic-Polar) model [**?**] goes as follows: Every amino acid is represented by a single "bead". To further simplify things, we consider only two types of amino acids – H (red) for hydrophobic, P (blue) for polar (hydrophilic). H amino acids repel water, while P amino acids attract water. In our simplified model the energetic forces acting between the units are reduced to a single rule: H amino acids like to stick together, while P units are inert, neither attracting nor repelling. This is indeed a very simplified model.

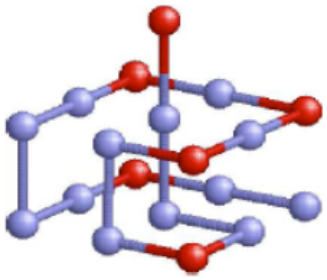


Figure 8: A 3-D lattice conformation. The blue beads represent polar amino acids and the red beads represent hydrophobic amino acids.

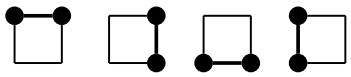


Figure 9: A self avoiding walk of size 1. The four walks are actually one and the same.

The H and P amino acids are placed at the grid points of the 2D lattice (see Figure 7). More sophisticated models use a 3-D grid (Figure 8) and/or models that allow diagonal walk. The peptide bonds are lines drawn on the grid Since the chain is confined to the lattice, the number of possible conformations is severely limited.

Given this representation, we can ask ourselves two related questions:

1. Given an n amino acid chain of H's and P's – How many sequences are there?
2. Given a specific sequence of H's and P's – How many conformations are there?

The number of possible sequences grows exponentially with n. Every position on the chain can be either H or P, but unlike real proteins, the model does not have directions, so for example, sequences like HP and PH are treated the same. Practically then, the number of sequences is smaller than 2^n. The second question requires some thought: The number of conformations translates to the number of self-avoiding walks on the 2D lattice. A self-avoiding walk is a path on the 2D grid that does not cross the same grid point more than once So, let us enumerate: How many shortest self-avoiding walks of length n-1 are there on the 2D grid?

The number of walks in general is 4^{n-1}, since at every step we can go in one of four possible directions: Up, down, left, right. However, we are interested in self avoiding walks. Moreover, many of these walks are symmetric. For example, for a path of length 1 (two amino acids), we have four possible walks that are trivially self-avoiding (Figure 9), but they are the same conformation rotated in space.

There are two distinct possible self-avoiding walks of size 2 (Figure 10).

The number of self avoiding walks grows quickly as the chain becomes longer. There are 2,900 self-avoiding walks of size 10 and 34,884,239 walks of size 20! (from: Sloane, N. J. A. Sequences A046170, and A046171 in "The On-Line Encyclopedia of Integer Sequences). The problem is still NP-Complete even on this simplified model, but the search space is greatly reduced [**?**]. We can sample the conformational space efficiently using Monte Carlo, but first we have to ask ourselves how we evaluate lattice based conformations.

Figure 10: The two self avoiding walks of size 2.

Evaluating Lattice Based Models Previously we asked ourselves: What makes a fold good? According to the thermodynamics hypothesis, a good fold has a Low Gibbs free energy (G = E - TS)

How do we model the free energy in such a simplified system? Accurately estimating free energy may be a challenge, but we can use an approximation: We can count the number of H-H contacts (the dashed lines in Figure 7) and use it as our scoring function. This should reflect the tendency of hydrophobic amino acids to go towards the core of the protein to avoid water. It is, of course, a very crude approximation. Polar amino acids don't count at all, but it will do for the sake of this simple example. Now, let us explore the conformational space.

Monte Carlo to Sample the Conformational Space As we saw, on long amino acid chains, enumeration is impractical, even with a simple model. Monte Carlo could be used to compute self-avoiding walks on the lattice, such that no two amino acids occupy the same lattice point. Given an HP chain of length N, do the following, starting with a random self-avoiding walk on the $N \times N$ grid:

For i = 1 to N_{cycles}

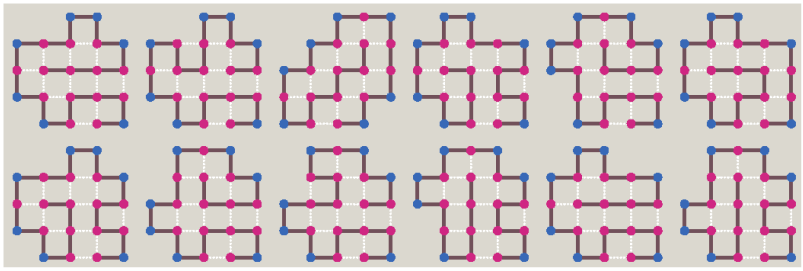
- Propose a move to obtain a new self-avoiding walk from the current one
- Estimate the score of the proposed walk with the ranking H-H score and compare it to the score of the current walk
- If the difference in score meets the Metropolis criterion as described above, accept the new walk and resume the next cycle from i

Figure 11 (a) shows the 12 of the 107 most stable folds of 80 21 amino acid sequences. As seen, the folds are compact and the red H's form stabilizing contacts (dotted white lines) when nearest neighbors blue P beads have no interactions, since they are not counted in the scoring function. Figure 11 (b) shows five possible folds of size 21, selected at random. As seen, random structures tend to be much less compact.

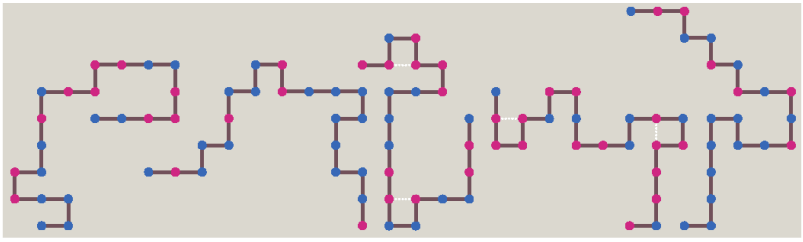
Finding Best 2D Lattice Folds of HP sequences Obviously, lattice based models do not give us an accurate, realistic view of a protein structure, but they can give us a lot of insight about the topology and arrangement of the full structure.

4 Off Lattice Models

Off lattice models give a more accurate and realistic representation of a protein structure, but they are more time consuming. Our degrees of freedom are usually the ϕ and ψ backbone dihedral angles. The scoring function is also more complex and should embody the physical and chemical interactions



(a)



(b)

Figure 11: (a) 12 out of the most stable folds of 21 amino acid sequences. (b) Five random folds of 21 amino acid sequences.

between the atoms. Before we talk about protein folding, let us demonstrate the concept using a simpler sub-problem: The secondary structure prediction.

5 Secondary Structure Prediction

This is a sub-problem of protein folding. The goal is to find out which segments of the proteins form α helices, which segment form β sheets and which form loops, coils or other secondary structure elements. As expected, it is an easier problem than predicting the tertiary structure. Further increase in number of deposited structures allows more accurate estimation of amino-acid composition of secondary structures. In parallel, the development of sophisticated machine learning techniques allow us to utilize the data for more efficient prediction. Many well established method perform secondary structure prediction quite successfully.

To measure secondary structure prediction accuracy we use the Q_3 score:

$$Q_3 = \frac{\text{Number of residues correctly predicted}}{\text{Total number of residues in protein}} * 100$$

Random prediction that follows the observed frequency of alpha-helices (39%), beta-strands (23%), and coils (38%) will give an average Q_3 accuracy of around 35%. First-generation secondary methods give 50%-56% accuracy, Second-generation methods: 70% accuracy, while state-of-the-art (current) methods produce 70%-80% accuracy.

Newer methods take advantage of multiple sequence alignment and evolutionary information to achieve higher accuracy. The assignment of secondary structure is a typical annotation problem

that can be addressed with various machine learning techniques. The most advanced methods for secondary structure prediction use Hidden Markov Models (HMM), Artificial Neural Networks (ANN), Support Vector Machine (SVM) and other machine learning techniques. All approaches capture key amino-acid level signals present in alpha-helices and beta-strands Since coils, loops, turns do not have such well-defined signals, they are usually predicted as "other" and are more difficult to pin down.

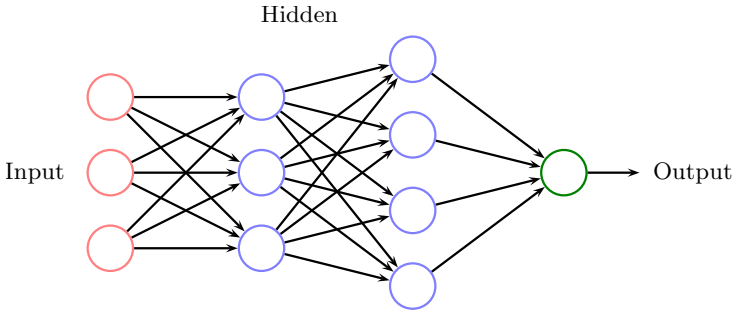
The Performance of Secondary Structure Prediction Methods Over the years there has been a significant improvement in the prediction accuracy, as the following summary of popular methods suggests.

- First-generation (three representative methods) – 50%-56% accuracy:
  1. Rules manually derived from known native structures of proteins Example: Lim et al. with accuracy 50%
  2. Automated statistics on amino-acid composition and neighbor effects Example: Chou-Fasman et al. with accuracy 53%
  3. Statistics on composition taken on 17-residue windows [-8, ?, +8], using a statistical framework to predict the secondary structure of the middle '?' residue Example: GOR method with accuracy 56%
- Second-generation helped by increase in deposited structures and the use of MSA to detect similar sequences with similar structures (two representative methods) - 70% accuracy:
  1. MSA information combined with HMMs, neural networks, SVMs Example: PHD with accuracy 70.8%
  2. k-nearest information (k= 50-100 window) gathered from a database on a voting principle Example: NNSP with accuracy 72.2%

In what follows we will show some examples of state-of-the-art secondary structure prediction methods.

5.1 Artificial Neural Network

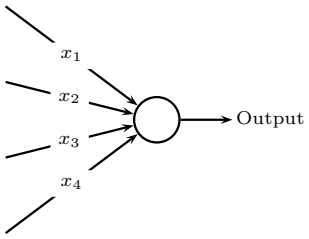
Artificial Neural Networks (ANN) are computational models inspired by the biological neural networks that constitute animal brains. They have been applied successfully for classification problems and pattern recognition. The network is represented as a weighted directed graph. The graph has a layered structure: An input layer, one or more "hidden layers" and an output layer.



Every node represents a neuron and every edge represents a connection (synapse) between neurons. Different layers may perform different functions on their inputs. Signals travel from the input layer, to the last output layer, possibly after traversing the layers multiple times. The input layer receives input from the outside. The output layer transmits output to the outside. The hidden layers are not connected directly to the outside, only to other layers.

The ANN receives input from the outside in the form of a vector. These inputs are designated as $x(n)$ for n inputs. Each input is multiplied by its edge weight. The weights represent the strength of the interconnection between neurons inside the network.

To understand how it works exactly, let us first look at what is perhaps the simplest ANN, a *perceptron*. A perceptron takes binary inputs, $x_1 \dots x_n$ and produces a single binary output. The inputs are weighted by real numbers: $w_1 \dots w_n$, scaling the importance of each input to the output. Overall, the input is a weighted sum $\sum_1^n w_i x_i$.



The output is either 0 or 1, depending on whether $\sum_1^n w_i x_i$ is below or above a given threshold. respectively:

$$Output = \begin{cases} 0, & \text{if } \sum_1^n w_i x_i < threshold \\ 1, & \text{otherwise} \end{cases}$$

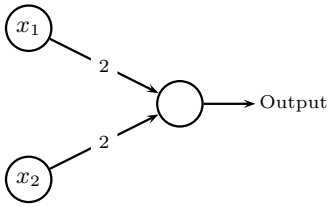
This is called a step function. You can think about it as a very simple decision making device, weighing up evidence from the input neurons.

Here is a simple example, a perceptron with two input neurons that calculates logical OR:

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

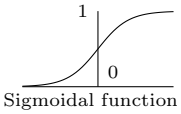
The activation function is:

$$Output = \begin{cases} 0, & \text{if } \sum_1^n w_i x_i < 2 \\ 1, & \text{otherwise} \end{cases}$$



The input neurons are binary, in this case, and the weights are 2. It is easy to see that the output is 2 if and only if both inputs are 0, and 1 otherwise.

To make the prediction more sophisticated, the step function from above is often replaced by a sigmoidal function with a smoother threshold: $Output = \frac{1}{1+e^{-x}}$.



Sigmoidal function

We can use this simple perceptron model to build increasingly complex networks. Each of the perceptrons in a given layer makes a decision based on the input from the previous layer. This way, each perceptron in a deeper layer can make decisions at a more complex and more abstract level than the perceptrons in a previous layer, And even more complex decisions can be made by the perceptron in the next layer. This way, a many-layer network of perceptrons can engage in sophisticated decision making.

There are many types of neural networks.

Definition 2 *feed forward network A feed forward network is a network where the output is only propagated in one direction: From the input to the output.*

This is the simplest neural network model. There are no feedback connections in which outputs of the model are fed back into itself.

Definition 3 *backpropagation network A backpropagation network is a network where an output can be fed back through the network in order to minimize the output error.*

Here, arbitrary weights are initially assigned and the output values are compared with the correct answer (target output) to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. The process continues for a pre-defined number of rounds or until the output converges to a good enough value with a small error.

ANN for Secondary Structure Prediction ANN is an example of a *discriminative model*: We build a model that separates between various classes or groups. One of the earliest methods for secondary structure prediction dates back to 1989 [**?**]. The input was a set of 62 proteins. 48 were used as the training set, and the remaining 14 were the test set. The network consisted of one input layer, a single hidden layer and an output layer, as shown in Figure 12.

The input layer is a sliding window on the amino acid sequence. The window size is 17, where the prediction is made for the central residue in the window. Each amino acid at each window

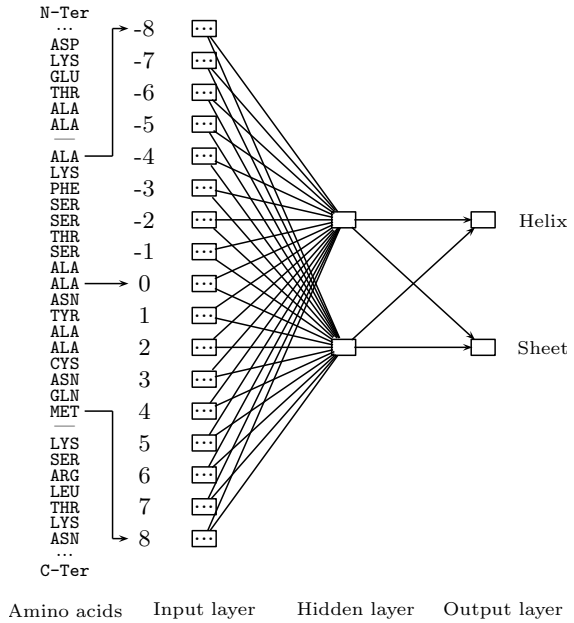


Figure 12: The architecture of a Neural Network for secondary structure prediction [**?**].

position is encoded by a group of 21 inputs, one for each possible amino acid type and one to provide a null input when the window overlaps with the N- or C- terminus. In each group of 21 inputs, the input corresponding to the amino acid type at that window position is set to 1 and all other inputs are set to 0. Thus, the input layer consists of 17 groups of 21 inputs each, and for any given 17 amino acid window, 17 network inputs are set to 1 and the rest are set to 0.

The hidden layer consists of two units. The output layer also consists of two units. Secondary structure is encoded in these output units as follows: $(1,0)$ = helix, $(0,1)$ = sheet, and $(0,0)$ = coil. Actual computed output values are in the range $0.0 - 1.0$ and are converted to predictions with the use of a threshold t.

Helix is assigned to any group of four or more contiguous residues having helix output values greater than sheet outputs and greater than t. β-Strand is assigned to any group of two or more contiguous residues, having sheet output values greater than helix outputs and greater than t. Residues not assigned to helices or sheets are assigned to coil. The value of t is adjusted by maximizing the accuracy of secondary structure assignment for the training set.

PSIPRED: A later method, still considered among the state-of-the-art, is PSIPRED [**?**]. PSIPRED uses PSSM (Position-specific scoring matrices) based on sequence profiles. The PSSM is obtained from PSI-BLAST on a custom-made sequence databank and used as input to the neural network. The matrix has $20 \times M$ entries (M is the size of the target sequence) Each entry is the log-likelihood of this particular substitution in the template based on the weighted average of the BLOSUM62 matrix. The profile matrix elements are scaled to the $[0 \dots 1]$ range using the logistic function $\frac{1}{1+e^{-x}}$, where x is the raw value. The method uses a feed-forward network with a single hidden layer. A window of 15 amino acids was found to be optimal for prediction, producing a Q_3 score of

80.1%. The input layer has $15 * 21 = 315$ units (an extra unit indicates the N- or C- terminus). A hidden layer has 75 units and another three units for the output layer for the prediction of helix, strand or coil.

A second network is used to filter successive outputs from the main network. That layer had 60 input units (15 groups of four, one extra unit again indicates a terninus). The hidden layer has 60 units.

5.2 Hidden Markov Models

A `Markov Model` is a statistical model used to model randomly changing systems. It assumes that future states of the system depend only on the current one, and not any past ones. A `Markov Chain` is a model that describes the system using a random variable that changes over time, and its distribution depends only on the state preceding it. More formally, we are given the following:

- A set of states $\{S_1, S_2, ..., S_N\}$
- A set of transition probabilities $a_{ij} = P(S_i|S_j)$
- A set of initial probabilities $\pi_i = P(S_i)$ for every i

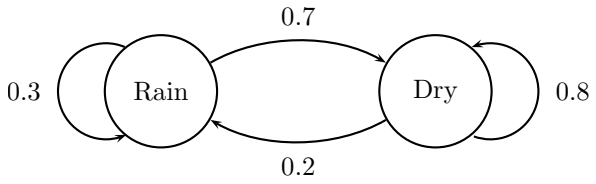
In addition, we assume that every state depends only on the previous state: $P(S_{ik}|S_{i1}, S_{i2}, ..., S_{ik-1}) = P(S_{ik}|S_{ik-1})$

Here is a simple example of a Markov Model, modeling the weather: Given two states, Rain and Dry, with the following transition probabilities:

From \ To	Rain	Dry
Rain	0.3	0.7
Dry	0.2	0.8

In addition, given the following transitional probabilities: $P(Rain) = 0.4$ and $P(Dry) = 0.6$.

Here is an illustration of the transition probabilities:



Let us calculate the probability of a state sequence as follows:

$P(S_{i1}, S_{i2}, ..., S_{ik-1}, S_{ik}) = P(S_{ik}|S_{i1}, S_{i2}, ..., Sik - 1)P(S_{i1}, S_{i2}, ..., S_{ik-1})$.

The Markov chain property states that the above is equal to:

$P(S_{ik}|S_{ik-1})P(S_{i1}, S_{i2}, ..., S_{ik-1}) = P(S_{ik}|S_{ik-1})P(S_{ik-1}|S_{ik-2})...P(S_{i2}|S_{i2})P(S_{i1})$.

In our example, if we want to calculate the probability of the sequence {Dry, Rain, Rain, Dry}, we calculate the following:

$P(\{Dry, Rain, Rain, Dry\}) = P(Dry|Rain)P(Rain|Rain)P(Rain|Dry)P(Dry) = 0.7 * 0.3 * 0.2 * 0.6 = 0.0252$.

A `Hidden Markov Model` (HMM) is a Markov model where not all states are observable. In other words, the rules that produce the Markov chains are not known or "hidden". The rules include the probability for a certain observation for a certain state transition, given the state of the model at a certain time.

We have the following properties, just like before:

- A set of states $\{S_1, S_2, ..., S_N\}$
- A set of transition probabilities $a_{ij} = P(S_i|S_j)$
- A set of initial probabilities $\pi_i = P(S_i)$ for every i

However, states are not visible, so in addition we get the following:

- Each state randomly generates one of M observations (or visible states) $\{V_1, V_2, ..., V_M\}$
- A set of observation probabilities: $b_i(V_m) = P(V_m|S_i)$. These are also called *emissions*.

The (HMM) method aims to solve the following problems:

1. given the model, find the probability of the observations.
2. given the model and the observations, find the most likely state transition trajectory.
3. maximize either 1 or 2 by adjusting the model's parameters.

Let us use the previous example to demonstrate HMMs: Say we now have two *observations*, {Rain, Dry} and two (invisible) *states*, {Low, High} (atmospheric pressure). In other words, we can observe whether it is rainy or dry, but we cannot directly tell what the atmospheric pressure is. The only way to recover the most likely atmospheric pressure is through the observations and the set of probabilities. We assume that the data observed is not the actual state of the model, but is instead generated by the underlying hidden states.

The transition probabilities are as follows:

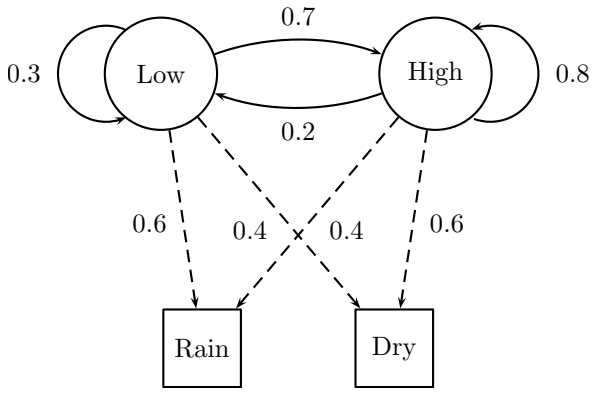
From \ To	Low	High
Low	0.3	0.7
High	0.2	0.8

The observation probabilities are as follows:

From \ To	Rain	Dry
Low	0.6	0.4
High	0.4	0.6

The initial probabilities are: $P(Low) = 0.4$ and $P(High) = 0.6$.

The graphical representation of the HMM is as follows:



If we are given a sequence of observations, say {Dry, Rain}, and want to calculate its probability, we need to account for all the possible hidden state sequences: P({Dry,Rain}) = P({Dry,Rain} , {Low,Low}) + P({Dry,Rain} , {Low,High}) + P({Dry,Rain} , {High,Low}) + P({Dry,Rain} , {High,High})

Every term can be calculated given the transition probabilities and the Markov chain properties:

P({Dry,Rain} , {Low,Low})=P({Dry,Rain} — {Low,Low}) P({Low,Low})

Now, based on the Markov chain property for the hidden states, the last term can be expressed as:

P({Low,Low}) = P(Low—Low)P(Low)

Additionally, the value of the observed variable depends only on the value of the hidden state at that time. Therefore, the entire formula can be expressed as:

P({Dry,Rain} , {Low,Low})=P(Dry—Low)P(Rain—Low) P(Low)P(Low—Low)= 0.4*0.4*0.6*0.4*0.3 = 0.01152

The other probabilities can be calculated in a similar manner.

Using HMM for Secondary Structure Prediction: HMM is an example of a *generative model*: we learn a model based on specific signals and see how well the model explains (classifies/annotates) the input queries. The problem of predicting secondary structure elements can be modeled as an HMM such that the observations are the amino acids along the sequence and the hidden process is the secondary structure [**?**]. The assumption is that secondary structure can be modeled by Markov chain and the amino-acids, which are the observed states, are independent of each other, conditionally to the hidden process, which is the secondary structure composition.

The simplest HMM models every secondary structure by a single state. The parameters of the model are the transition and emission probabilities. More complex models assign several hidden states per secondary structure. Model parameters are then estimated from available data.

One of the state-of-the-art methods, OSS-HMM (Optimal Secondary Structure prediction Hidden Markov Model) [**?**] calculates secondary structure elements with 75.5% accuracy.

The outline of the algorithm is as follows: Let n_H, n_b and n_c be the number of hidden states that model α-helices, β-strands and coils, respectively. The optimal model selection is done in three steps:

1. At first, we set $n_H = n_b = n_c = n$ and estimate models with n running from 1 to 75. The Q_3 score indicates that 10 to 15 states are sufficient to obtain good prediction without overfitting. Eventually, $n = 14$ was selected based on the Bayesian Information Criterion (BIC, see below) of the model.

2. Models were thus estimated with:

   (a) $n_H = 1$ to 20 and $n_b = n_c = 1$,

   (b) $n_b = 1$ to 15 and $n_H = n_c = 1$

   (c) $n_c = 1$ to 15 and $n_H = n_b = 1$.

   The BIC selected $n_H = 15$, $n_b = 8$ $n_c = 9$

3. All the architectures were tested with n_H varying from 12 to 16, n_b from 6 to 10 and n_c from 3 to 13. The BIC selected the optimal model having 36 states with $n_H = 15$, $n_b = 9$ and $n_c = 12$.

   The models are built without a priori knowledge. All transition probabilities are initially set to $\frac{1}{N}$, where N is the total number of hidden states. Initial emission parameters are selected at random. Different starting point are used and the model with the best likelihood is selected during the EM estimation. The number of starting points is 10 or 100, depending on the size of the model.

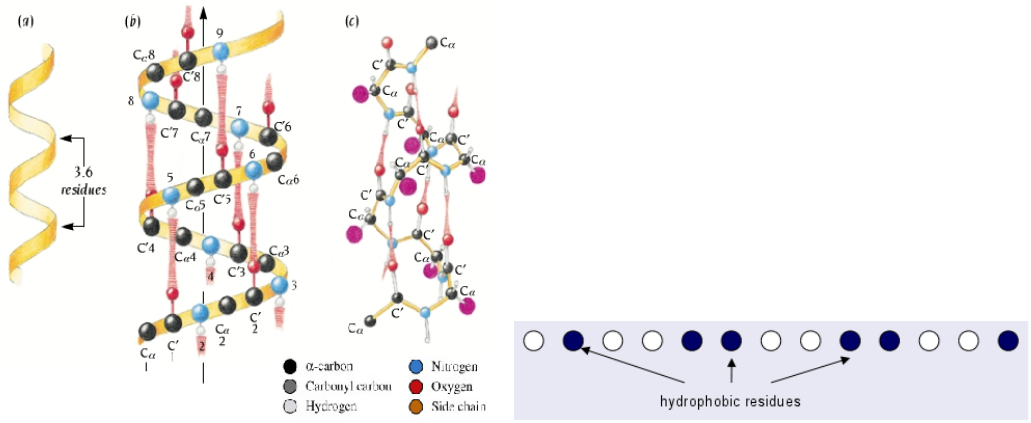
   Model Estimation: Three criteria are used for the selection of the optimal model:

   (a) Q_3 as described above

   (b) The Bayesian Information Criterion (BIC) ensures the best compromise between the goodness-of-fit of the HMM and a reasonable number of parameters. It is defined as: $BIC = \log L - 0.5 \times k \times \log(N)$, where $\log L$ denotes the log-likelihood of the learning data under the trained model, k is the number of independent model parameters and N is the size of the training set.

   (c) The statistical distance between two models. The distance D_s between models M_1 and M_2 is given by:
   $D_s(M_1, M_2) = \frac{D(M_1,M_2)+D(M_2,M_1)}{2}$, and
   $D(M_1, M_2) = \frac{1}{T}|\log L(O^{(2)}|M_1) - \log L(O^{(2)}|M_2)$,
   where $O^{(2)}$ is a sequence of length T generated by model M_2 and $\log L(O^{(2)}|M_i)$ is the log-likelihood of $O^{(2)}$ under model M_i.

   The selection process first considers models with equal numbers of hidden states. The three criteria are used to select models with a limited number of hidden states. Then, models are considered where the number of states are set to one for two of the secondary structure classes, and increase for the remaining classes. This defines the model size range that needs to be explored for each structural class: 12 to 16 states for helices, 6 to 10 states for strands and 5



to 13 states for coil. All these models are generated and evaluated, and the optimal model is selected as mentioned below.

The secondary structure labels were assigned using the DSSP program [**?**] and the learning was done using the Expectation-Maximization (EM) algorithm [**?**].

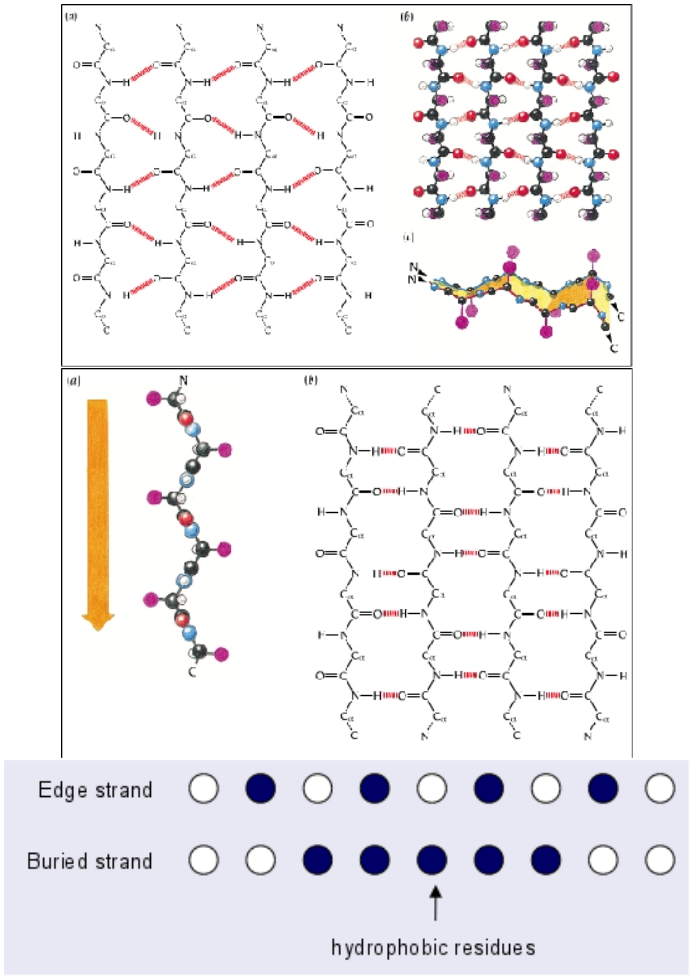
Transition states: All transitions between hidden states are initially allowed. However, many transitions in the final model are estimated to have probability zero. In fact, only 36% of potential transitions remain within the helix box, 57% within the strand box and 68% in the coil box. Thus, we can conclude that the paths within the helix box are more constrained than in the strand box, despite the fact that there are more helix states. The final model has 448 non-null transitions (out of the possible $36^2 = 1296$), of which 89 have a probability greater than 0.1, for a total of 1096 free parameters.

Amino acid preferences: An examination of amino acid preference scores shows that hidden states specifically tend to avoid particular residues rather than favor other ones. This seems to denote a kind of "negative design", where there is a stronger constraint not to include particular residues, than to have some others.

Sequence Signals on Different Secondary Structures: α-Helices have 3.6 residues per turn, and their lengths are typically 5-40 residues (10 residues on the average), which gives about 3 turns: Helices also tend to be amphipathic: The side that faces the solvent tends to have hydrophilic amino acids, and the side that faces the protein interior tends to be hydrophobic.

β-strands are typically shorter than α-helices. In fact, they can be as short as 2-3 residues

Composition signal for specific amino-acids include knowledge about the propensity of some residues to appear in secondary structure elements. For example, Proline residues do not occur in middle segments, but can appear in the first two positions of the helices. There are often specific pattern of hydrophobic and polar residues. Also, studies show that amino-acid preferences vary according to the position on the helix – beginning, middle or end. The composition signals for β-sheets are typically not as strong signals as in α-helices. There are usually specific pattern of



hydrophobic and polar residues: The hydrophobic nature of each strand is different depending on whether or not it is internal or external to the sheet.

Coils do not have well-defined geometric structures. They're usually extended or loops/turns These regions are highly variable when subjected to multiple sequence alignment. Coils usually contain a high proportion of small polar residues: they tend to be at the surface/exposed side of the protein, and not be buried inside the hydrophobic core Coils also tend to be more flexible in protein structures. Computing conformations for coils is considered a mini-version of the general structure prediction problem (see below).

6 Tertiary Structure Prediction

6.1 Homology Modeling

Homology modeling or comparative modeling is modeling of the unknown based on comparison to what is known. In the context of modeling or computing a protein structure, we assume that similar sequences adopt similar structures. In other words, if s_x is the structure assumed by a sequence x of amino acids, the underlying assumption is that the structure is a function of the sequence: So, $s_x = f(x)$. The function f encodes how the sequence x determines the structure s_x. We may not know what f is, but it exists. So, given another protein of sequence y and *known* structure s_y, we can infer: IF $x \approx y$ THEN $s_x \approx s_y$ It is important that x and y be similar enough.

The important question is how similar is similar enough? When can we infer that the structure assumed by the sequence x is similar to that assumed by the sequence y? The surprising answer – the sequences do not have to be nearly identical or even close, to assume very similar structures. Statistical analysis of sequences with known structure reveals that sequences with at least 40-50% sequence identity assume very similar structures. While the exact numbers change from one study to another and from one protein family to another, most studies agree that the minimum sequence identity that can be used for comparative modeling is as low as 30%. Thinking again, it is not so surprising. It is well known that structures of proteins in a given functional family are more conserved than their sequences and in general, structure is conserved better than sequence.

Figure 13 shows a schematic representation of the three zones of sequence/structure similarity:

A: Less than 30% sequence identity usually does not allow for a reliable sequence based homology modeling.

B: Sequence identity between 30% and 50% sequence identity: Some structural variations are possible, especially in loop regions. This is the "twilight zone"

C: Sequence identity of 50% and higher: The structural similarity is expected to be very high. Some variations are still possible, especially in loops. In any case, careful model validation is advisable.

So, if we cannot yield a high-resolution structure, comparative modeling can at least yield the overall fold for a sequence with an unknown structure for which a similar sequence with a known structure exists. Currently, comparative modeling is both faster and more accurate than other structural prediction methods, as long as we can find a known structure with high sequence identity. Before we move on to describing homology modeling in detail, here is some terminology:

- The protein of unknown structure is the *query* or the *target*.
- The protein of known structure whose sequence is similar to that of the target is the *template*
- The process of inferring the coordinates for the target is called *model building*

6.1.1 A Simplistic View of Comparative Modeling

Comparative modeling builds the model, completes it, refines it, and then evaluates it.

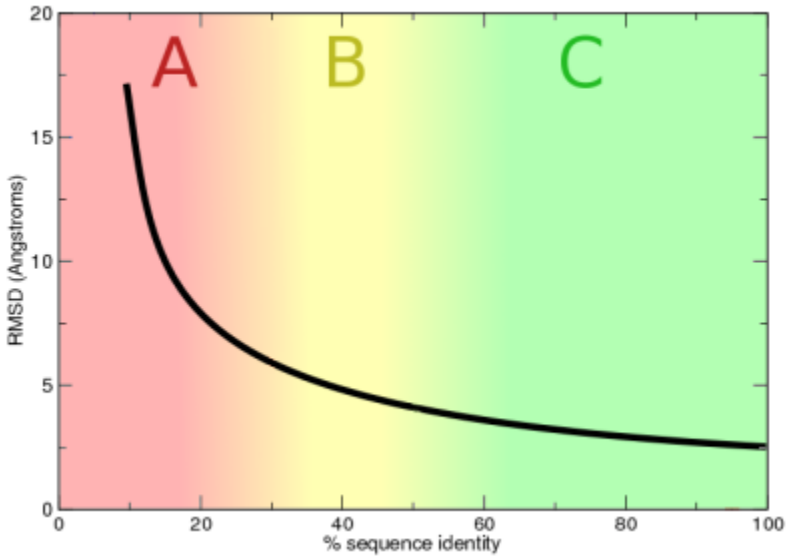


Figure 13: A schematic representation of the relationship between sequence and structural similarity. The three zones show the possible relationships between sequence and structural similarity.

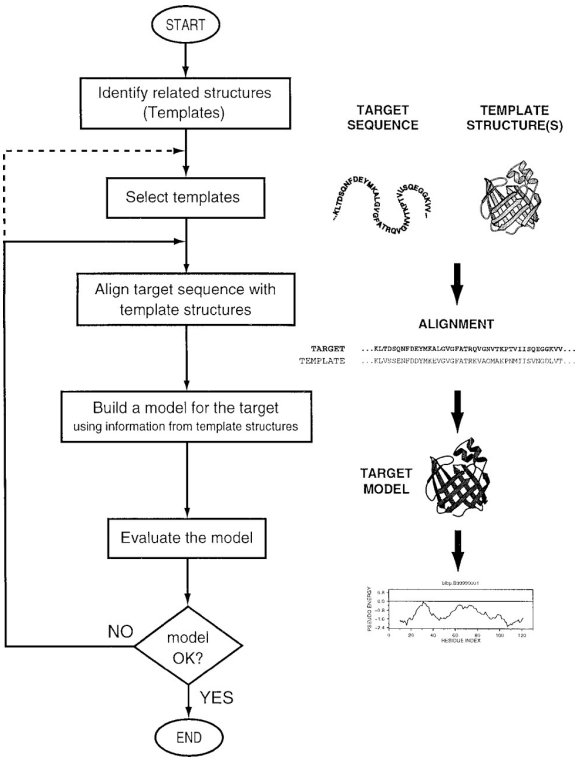


Figure 14: A basic flowchart of the homology modeling process.

6.1.2 Basic Steps of Homology Modeling

Before going into details about the homology modeling process, here is a summary the basic steps:

1. Query a database of protein sequences with known structures with the target sequence, focusing on those with $\geq 30\%$ sequence identity to the target sequence
2. Align obtained sequences to target to choose templates
3. Identify structurally conserved (SC) and variable (SV) regions
4. Generate coordinates for the core region of the target
5. Complete the structure of the target
    - generate coordinates for loop regions
    - generate coordinates for side-chains
6. Refine the completed structure using energy minimization
7. Validate/evaluate completed structure

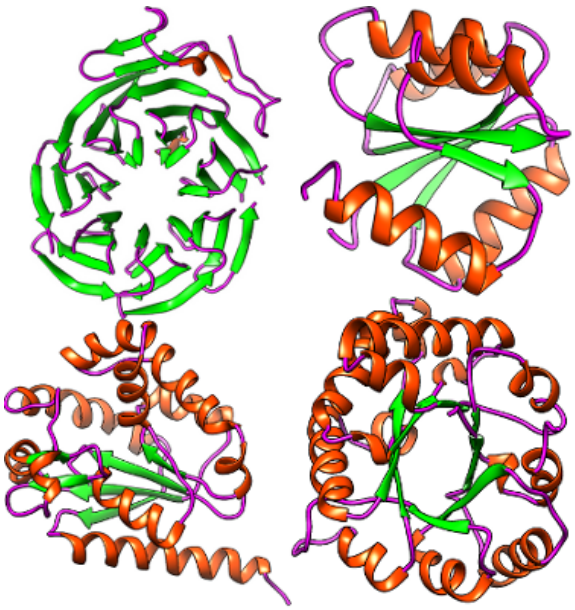
Step 1 – Query PDB: Our goal: Find a template or templates with high sequence identity. To find high homology sequences, we align our query sequence against a sequence database. The most common way to do it is to use a stand-alone software or online server. Many such servers and programs are available, and for the most part they query the BLAST database (http://www.ncbi.nlm.nih.gov/BLAST). It is always good to search for many possible templates. Multiple sequence alignment can be done with PSI-BLAST and CLUSTAL.

What is a good template? If possible, we want a good template to be closest to the target in terms of subfamilies This means that high overall sequence similarity is needed. The template environment like pH, ligands, etc., should be the same as that of the target The quality of the experimentally-available template structure - the resolution, R-factor, etc. – should be high. When choosing a template for a protein-ligand model, it is preferred that the template have the same ligand. When modeling an active site – a high resolution template structure with ligand is important. For more about protein-ligand binding and active sites, see chapter 6.

- Pairwise sequence alignment: BLAST, FASTA, WU-BLAST, SSEARCH, and more
- Available as web servers and standalone software
- Basic functionality needed: compare target sequence with sequences in the PDB (or any other comprehensive structural database)
- BLAST scans the sequence for 3-letter words (wmers, where w = 3) and expands alignments from 3-mers
- Statistically significant alignments are hits
- Templates are hits with no lower than 30% sequence identity

PRTEINSEQENCEPRTEINSEQUENCEPRTEINSEQNCEQWERYTRASDFHG
TREWQIYPASDFGHKLMCNASQERWWPRETWQLKHGFDSADAMNCVCNQWER
GFDHSDASFWERQWK

Query Sequence



PDB

Figure 15: Description of searching for a sequence that matches a database of protein structures.

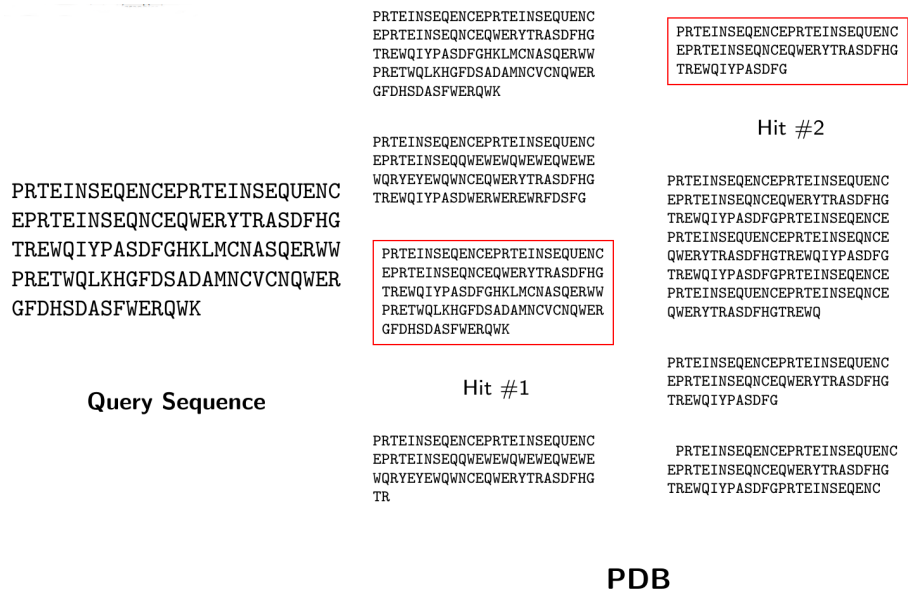


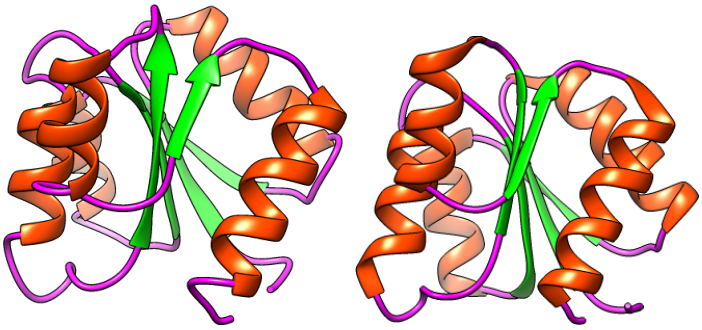
Figure 16: Searching for matches in protein sequences.

	G	E	N	E	T	I	C	S
G	10	0	0	0	0	0	0	0
E	0	10	0	10	0	0	0	0
N	0	0	10	0	0	0	0	0
E	0	0	0	10	0	0	0	0
S	0	0	0	0	0	0	0	10
I	0	0	0	0	0	10	0	0
S	0	0	0	0	0	0	0	10

	G	E	N	E	T	I	C	S
G	60	40	30	20	20	0	10	0
E	40	50	30	30	20	0	10	0
N	30	20	40	0	0	0	10	0
E	20	20	20	30	20	10	10	0
S	20	20	20	20	20	0	10	10
I	10	10	10	10	10	20	10	0
S	0	0	0	0	0	0	0	10

Figure 17: Dynamic Programming for aligning two sequences.

```
Query    ACDEFGHIKLMNPQRST--FGHQWERT-----TYREWYEG
Hit #1   ASDEYAHLRILDPQRSTVAYAYE--KSFAPPGSFKWEYEA
Hit #2   MCDEYAHIRLMNPERSTVAGGHQWERT----GSFKEWYAA
```



(a) Hit 1 (b) Hit 2

Figure 18: Two hits obtained by sequence query.

Step 2 – Alignment Alignment is done using Dynamic programming, as in regular sequence alignment.

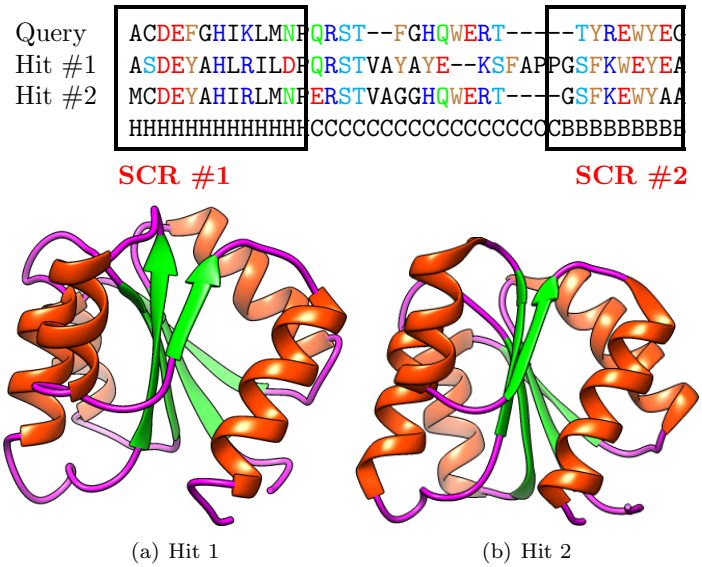
Global sequence alignment (Needleman-Wunsch) [**?**] can be used Notice: Alignment is the most crucial step, as comparative modeling can never recover from a bad alignment! Even a small error in the alignment can translate to a significant error in the reconstructed model Multiple sequence alignments (that also align the templates to one another) is often better than pairwise alignment. This is possible if there are multiple possible templates. Multiple sequence alignment methods include for example Clustal [**?**], Psi-BLAST [**?**].

Step 3 – Detect Structurally Conserved Regions (SCRs) SCRs correspond to the most stable structures or regions (usually in the interior/core) of the protein They also often correspond to sequence regions with the lowest level of gapping and highest level of sequence conservation. Usually, they are secondary structures. See Figure 19.

Step 3 – Detect Structurally Variable Regions (SVRs) SVRs correspond to the least stable or the most flexible regions (usually in the exterior/surface) of the protein. They usually correspond to sequence regions with the highest level of gapping and lowest level of sequence conservation – usually loops and turns.

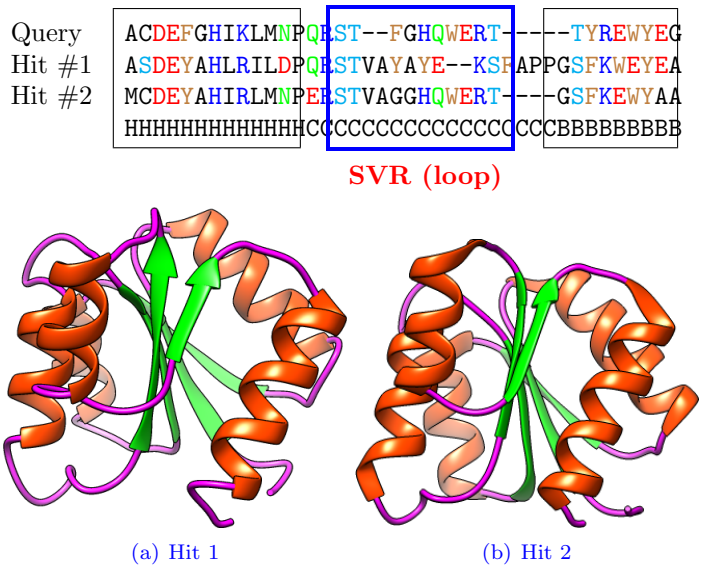
Step 4 – Threading At this stage the query sequence is fitted (threaded) onto the template structure.

- In positions where the query and template amino acids are identical, we just transfer all template atom coordinates (x, y, z) to the query protein (both backbone and side-chain atoms are identical).
- For similar but not identical amino acids, we transfer the backbone coordinates and replace side-chain atoms while respecting χ side chain angles.



(a) Hit 1 (b) Hit 2

Figure 19: The two hits with conserved regions



(a) Hit 1 (b) Hit 2

Figure 20: The two hits with conserved regions

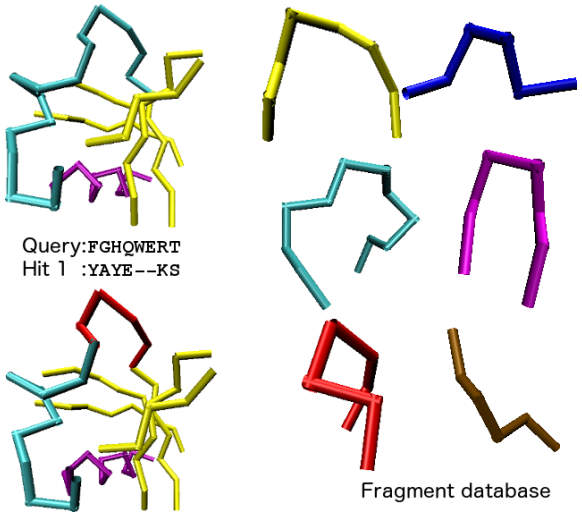


Figure 21: Illustration of Loop Modeling.

- For different amino acids, one can only transfer the backbone coordinates (x, y, z) to query sequence The side chains of different amino acids have to be built at a later stage, when completing the model. See stage 6 below.

Step 5 – Loop Modeling Loops often determine the functional specificity of a given protein framework and contribute to active and binding sites and loop modeling is therefore an active area of research. Modeling a loop is like a mini protein folding problem – loop sizes can vary from a few amino acids to very long loops, especially in membranes. Experimental data however, e.g. X-ray crystallography does not always provide structure models with all parts of the protein resolved at atomic resolution. Flexible parts of proteins, particularly loops, are often not resolved and hence, need to be modeled computationally. Basically two different approaches are employed:

1. Physics-based methods. The modeling process is regarded as a mini folding problem and loop conformation are produced by employing distance restraints to force the loop to the anchor positions. Subsequent minimization, heating and again minimization yields loop conformations with low energies. Ab-initio loop modeling is akin to minuscule folding, using similar methods – Monte Carlo, Monte Carlo with simulated annealing, MD, main chain dihedral angle search biased with the data from PDB, inverse kinematics (robotics) based, and others. More on inverse kinematics and robotics-based sampling – see next chapter.

2. Knowledge- based methods. The missing loop is built using homology modeling, thereby searching databases for fragments with the same sequence as the missing loop (see Figure 21).

Side Chain Modeling Side chain packing is critical especially for protein-protein and protein-ligand binding. Side chain modeling methods may predict the side-chain placement from similar structures, use steric and energetic considerations with conformational search algorithms, and sometimes combine rotamer libraries, which are datasets of energetically favorable rotamers (side chain

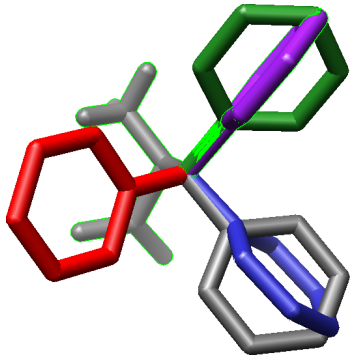


Figure 22: An example of five different side chain conformers (rotamers) of PHE.

conformers). Rotamer libraries have been created using statistical analysis of torsion angles of side chains of amino acids from structures in the PDB [**?**]. Figure 22 show an example of five different rotamers of Phenylalanine.

Two main effects in predicting side chains are:

- How it sits on top of the main chain(very critical)
- Continuous variation of side chain torsions – only 6% of the side chains vary more than ± 40 degrees from the rotamer libraries

Current techniques are able to predict side chains up to 1.5Å accuracy for a fixed (given) backbone conformations for the core residues, where the side chain positioning is mostly driven by VdW forces. Solvation and Hydrogen bond terms are very important in modeling exposed side chains

Many side-chain determination methods are available – SCWRL, SCAP, MODELLER, Insight II, WhatIf, SCREAM etc. [**?**, **?**, **?**].

- Evaluation of all three methods for backbone < 4Å lRMSD to native all work equally – 50% of $\chi 1$ and 35% of $\chi 2$ and $\chi 3$
- SCWRL – Decomposition of protein to non-interacting parts, collision free energy function. Fast, works quite well
- SCREAM – works well – accurate energy analysis – computationally intensive

Step 6 – Refinement Completed model may undergo a short energy minimization Physics-based or knowledge-based functions may be used The minimization may help remove steric clashes and improve favorable interactions in the completed model prior to the final evaluation of the built model for the target. Figure 23 shows a model created by Swiss-Model [**?**]

Comparative Modeling – Example

Evaluation of Model Given a predicted structure, there are several ways to evaluate the model. Here are some:

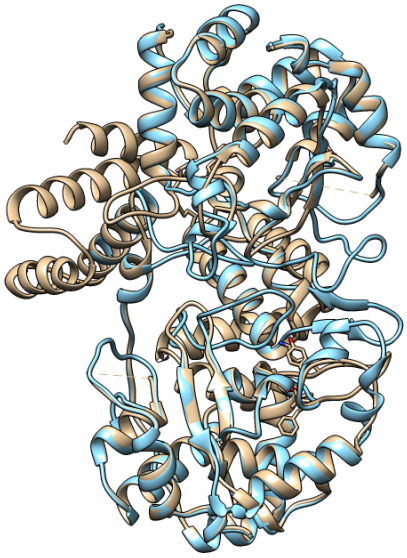


Figure 23: Beige – The structure of a dimeric Xer recombinase from archaea selected as template (PDB: 4a8e). Blue – Model, predicted by Swiss-Model [**?**]

- Ramachandran plot – allowed regions for backbone torsions
- Calculate the Hydrogen-bond network - use Quanta or WhatIf or MolProbity – normally calculated for heteroatoms with distance cutoff
- Identify hydrophobic residues on the surface
- Identify hydrophilic residues in the core – satisfied with salt bridges?
- Voids in the core are typically small two water cluster?

Comparative Modeling on the WWW

Prior to 1998, comparative modeling could only be done with commercial software or command-line freeware The process was time-consuming and labor-intensive The past few years has seen an explosion in automated web-based comparative modeling servers Now anyone can! (but you still have to know what you're doing...). The model in Figure 23 was done using a free online server, Swiss-Model. Swiss-Model [**?**] is a popular modeling server. The pipeline is made of the following steps:

1. **Input Data:** The input is a FASTA sequence or UniProtKB ID representing a sequence.
2. **Template Search:** Sequences are searched against a structural template library called STML using either BLAST or HHblits, a Hidden markov model based method to detect more remote homologues.
3. **Template Selection:** Multiple templates are selected and ranked according to the scoring functions described below.

4. **Model Building:** The modeling engines are called ProMod3 and OpenStructure. ProMod3 uses threading to match the sequence to the template as described above. For insertions and deletions it first searches for viable candidate fragments. If no suitable candidates are found, a Monte Carlo sampling is done. Side chains are being built using the Dunbrack's rotamer library and the SCWRL4 energy function.

5. **refinement** Energy minimization to resolve small clashes is performed using the CHARMM27 force field.

6. **Quaity Estimation:** Two quality estimates are used:

   - GMQE (Global Model Quality Estimation) is a quality estimation which combines properties from the alignment and the template search method. The score is a number between 0 and 1, reflecting the expected accuracy of a model built with that alignment and template and the coverage of the target. Higher numbers indicate higher reliability.
   - QMean score is a composite estimator based on different geometrical properties and provides both global (i.e. for the entire structure) and local (i.e. per residue) absolute quality estimates on the basis of one single model.

In addition, the server provides a quality assessment of the model using a Ramachandran plot, residue-residue contacts and per-residue QMean score, as shown in Figure 24.

6.2 Threading (Fold Recognition)

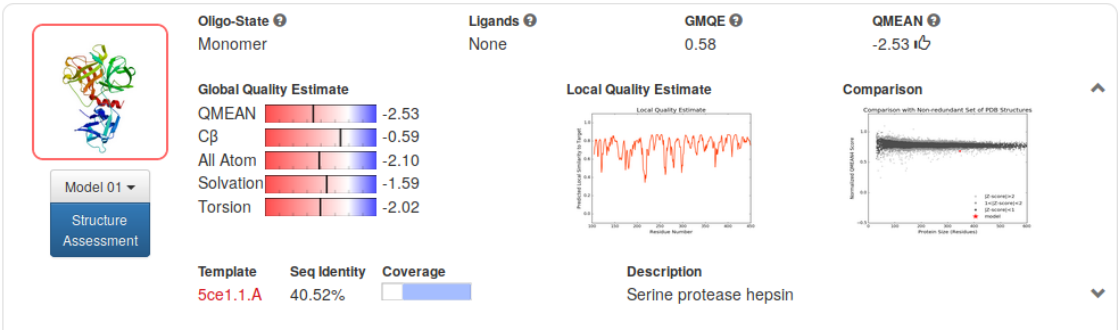
We may pause for a moment and ask ourselves – What are folds, anyway? Here is a citation from SCOP [**?**]:

- **Family:** Proteins are clustered together into families on the basis of one of two criteria that imply their having a common evolutionary origin: first, all proteins that have residue identities of 30% and greater; second, proteins with lower sequence identities but whose functions and structures are very similar; for example, globins with sequence identities of 15%.

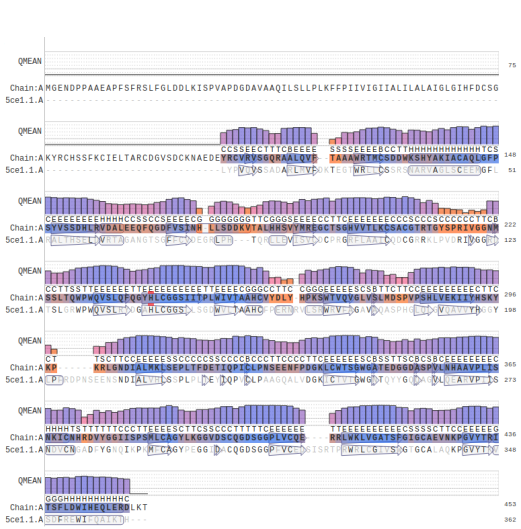
- **Superfamily:** Families whose proteins have low sequence identities but whose structures and, in many cases, functional features suggest that a common evolutionary origin is probable, are placed together in superfamilies; for example, the variable and constant domains of immunoglobulins.

- **Fold :** From SCOP: "Proteins have a common fold if they have the same major secondary structures in the same arrangement and with the same topological connections, though the length of regions can change. Structural similarities can arise just from the physics and chemistry of proteins favoring certain packing arrangements and chain topologies".

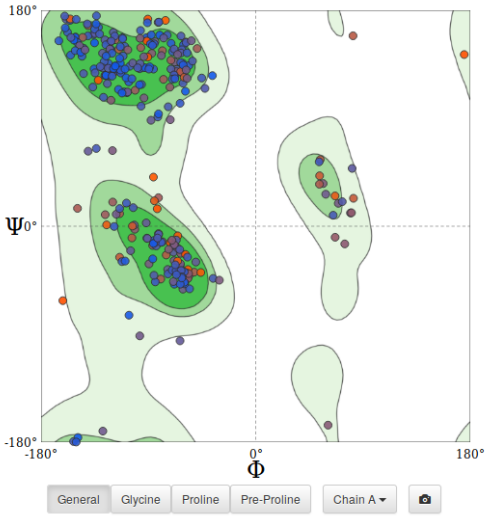
A large number of protein sequences adopt similar folds. Even when the sequence identity is lower than 25%. Such sequences are known as remote homologues Chothia and Lesk (1986) derived a general quantitative measure for measuring the relationship between sequence and structure similarity. As a rule of thumb, models and experimentally determined structures have shown that $> 30 - 35\%$ sequence identity between the model and the template generally gives reasonably good results. This 30% cutoff is often referred to as the minimum threshold for sequence-based structural modeling. Beyond 40% sequence identity, structural similarity is usually very high.



(a)

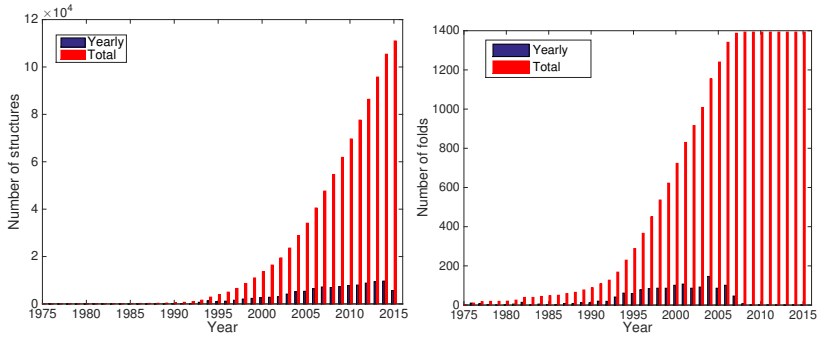


(b)



(c)

Figure 24: Quality measures for Swiss-Model: a. Overall evaluation. b. Per-residue Qmean score. c. Ramachandran plot



(a) (b)

Figure 25: The number of structures (a) and unique folds (b) deposited into the PDB. Source: http://www.rcsb.org.

Figure 26: Schematic illustration of threading.

So, proteins may have rather low sequence similarity and they may have no evolutionary or functional relationship, and yet they may share the same fold or even very similar 3-D structures! In other words – while the number of possible sequences is virtually unlimited, the number of folds is believed to be rather limited. As a matter of fact, we may have reached the maximum number of possible folds. Figure 25 shows that while the number of structures deposited into the PDB increases steadily, the number of unique folds has been much smaller and decreased sharply since around 2006. No unique fold has been detected since 2008 (source: PDB).

Threading, or fold recognition, is a structural modeling framework that relies exactly on the assumption that the environment of a particular amino acid is expected to be more conserved than the actual sequence identity of the amino acid. In other words, structure is conserved much better than sequence, and the overall fold of the protein is often not sensitive to sequence changes. This allows us to detect structural similarities that are not accompanied by detectable or significant sequence similarities.

A (simplistic) Overview of Threading

Threading algorithms follow this general scheme:

Step 1: Construction of template library from known folds

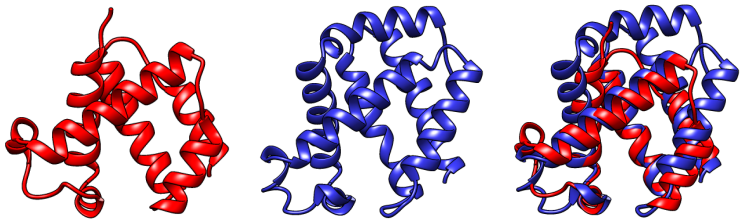
Step 2: Design of (energetic) scoring function

Step 3: Perform sequence-template alignment

Step 4: Template selection and model construction

Step 5: Model completion, refinement, and evaluation

See Figure 26 for a schematic flowchart:



(a) (b) (c)

Figure 27: Two hemoglobins with 18% sequence identity. a) mini hemoglobin from Cerebratulus lacteus (PDB: 1KR7). b) hemoglobin from F. Hepatica (PDB: 2VYW). c). The two structures superimposed. Despite the low sequence identity, The RMSD between the matching residues is 2.11Å.

This two structures in Figure 27 belong to the hemoglobin superfamily and have only 18% sequence identity. Homology modeling can no longer be applied here. However, these remote homologs have a similar fold! The Root mean square deviation (RMSD) between the aligned residues is 2.11Å. 103 amino acids are fully aligned. Not bad, considering that the smaller protein has 110 amino acids.

Threading: Problem Statement

Threading essentially approaches the following problem: Given a query sequence, find which fold "fits" best from a library of known folds This essential component of threading is often known as fold recognition (recognizing the best fitting fold among the library of available folds)

Sequence-structure Alignment

So, how do we align sequence and structure? Finding the best fold for a query sequence entails addressing the sequence-template alignment problem, which consists of two sub-problems:

1. Design of an effective scoring function to determine the "goodness" of a fit that results from an alignment (this is known as the threading energy function or potential)

2. Rapid alignment algorithm so that the sequence can be efficiently threaded to many folds during the search for the best one

Different methods spanning from "hard" to "soft" threading consider aligning sequence to sequence (like comparative modeling), sequence to structure, or sequence to contact environment.

Threading Energy Function

Sequence-template alignments are scored using a threading energy (objective) function. The function scores the compatibility between the query sequence of amino acids and their corresponding positions in a given template The objective function essentially scores compatibility using specifically-chosen parameters such as:

- Amino-acid preferences for solvent accessibility

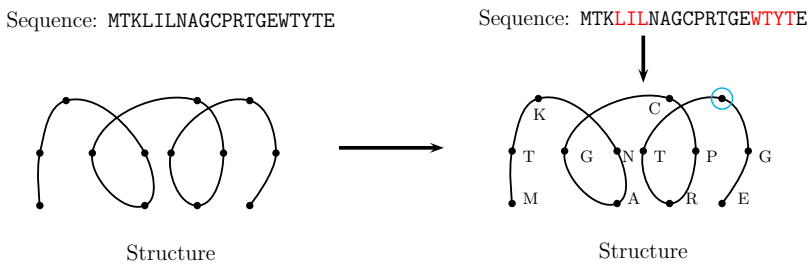


Figure 28: An illustration of a sequence-structure threading. The sequence is "threaded" onto a structure (right) and a scoring function is calculated. In this example some residues could not be aligned (depicted in red). A gap in the structure is circled in cyan.

- Amino-acid preferences for particular secondary structures
- Interactions between neighboring amino acids
- Inexpensive physics-based terms are also incorporated

Here is an example of a potential function used in Raptor [Xu et al.(2003)]: Our objective function is $E = E_p + E_s + E_m + E_g + E_{ss}$, where:

- E_p: how preferable to put two particular residues nearby (Pairwise potential)
- E_s: how well a residue fits a structural environment (Fitness score)
- E_m: sequence similarity between query and template proteins (Mutation score)
- E_g:alignment gap penalty
- E_{ss}: consistency with secondary structures:

Now, we have to minimize E to find the best sequence-template alignment (see Figure 28. There are three main approaches to the alignment sub-problem:

1. Sequence-sequence alignment (1D-1D) aligns a query sequence with template sequences.
2. Consider structural environment in addition to sequence (3D-1D)
3. Consider pairwise contacts in folds

Here are the main principles behind the three approaches:

Sequence - Sequence Alignment to Template: Essentially, this is similar to the process used in homology modeling. The advantage is that we can use simple dynamic programming methods to align the query sequence to the sequences of the templates, and guide the threading of the sequence into the structure. However, Templates may have low sequence similarity to query sequence and simple sequence alignment fails to consider interactions between neighboring amino acids It should be emphasized that there is no strict boundary between comparative modeling and threading in terms of methodology: rule of thumb is that when the alignment takes into consideration structural aspects (besides sequence aspects) and the templates are remote homologues, then we talk about threading rather than comparative modeling.

Consider structural environment in addition to sequence: Most successful threading algorithms consider the structural environment in addition to sequence. So instead of aligning a query sequence to a template sequence, these methods try to fit the query sequence to a *template structure*. This is called 3D-1D alignment. How do we align sequences and structures, then? There are several ways to do that.

Early methods try to convert the 3D structure into a string of variables that capture the 3D environment of the template structures. A well known threading algorithm by Lüthy, Bowie and Eisenberg [Threading] defined 18 environmental classes for each position in a template structure. First, every position divided into three sub-classes according to the secondary structure of the amino acid (α-helix, β-strand, or other). Then, each position is further divided into one of six sub-categories:

- **B1:** buried and hydrophobic environment
- **B2:** buried and moderately polar environment
- **B3:** and buried and polar environment
- **P1:** partially buried and moderately polar environment
- **P2:** partially buried and polar environment
- **E:** exposed to solvent

Tabulating Amino Acid Environments Each position in the template structure is mapped into one of the 18 possible environment classes (a vector of 18 descriptors). Different amino-acids prefer different environments as we have seen earlier, when discussing amino acid types. This preference is captured by compiling descriptors for each amino acid over structures in the PDB. The number of times an amino acid appears in a specific environment class is tabulated to obtain a frequency distribution. These frequencies are normalized for each amino acid to obtain probabilities of the form P(x, y), which is the probability of finding amino acid x in environment class y.

Each template structure is mapped to an environmental profile. Each position in the template is mapped by the sequence identity of the amino acid (find column) and the actual environment of that amino acid in the structure (find row) Each position is scored in this way – total profile score of the template is additive, so obtained by summing over the scores of the amino acids

Environment and Sequence Alignment: Aligning the query sequence to the profile of a specific template is similar to the traditional sequence-sequence alignment with DP: For each amino acid s_i in the query sequence, the cost of modeling it through amino-acid t_j in the template is either:

- that of "mutation" : environment cost provides this
- that of insertion: gap penalty incurred

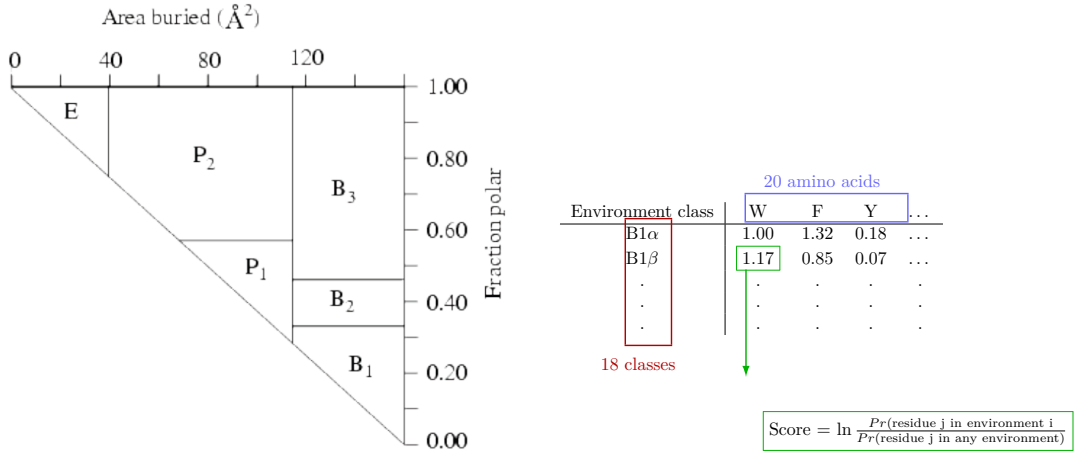
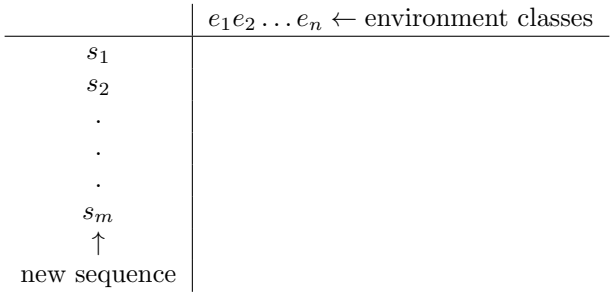


Figure 29: A sequence-structure profile alignment matrix.



Sequence to 3D Profile Alignment This method Provides good-quality models, since it considers not only sequence, but structural considerations encoded in the environment of an amino acid. By encoding structural features into a sequence-like profile, this method utilizes the highly efficient dynamic programming framework. On the other hand, amino acids are considered/threaded independently of one another. This is inherent in the additive score used in the dynamic programming formulation of the problem. To move away from this issue, the environment of each amino acid has to be considered as well.

6.2.1 Consider pairwise contacts in folds

Most successful threading methods fall in this category. Aligning a sequence to a structural profile fails to account for interactions between pairs (or more) of amino acids. One of the earlier examples of threading methods that considered pairwise contacts in folds is RAPTOR [Xu et al.(2003)] was one of the state of the art structural prediction tools. It uses PSI-BLAST [**?**] to find key positions of conserved regions and finds likely regions of secondary structure elements. The threading stage aligns the query sequence to the template using linear programming. The template is taken from a database of representative protein folds. Here is a description of the modeling, shown schematically in Figure 30:

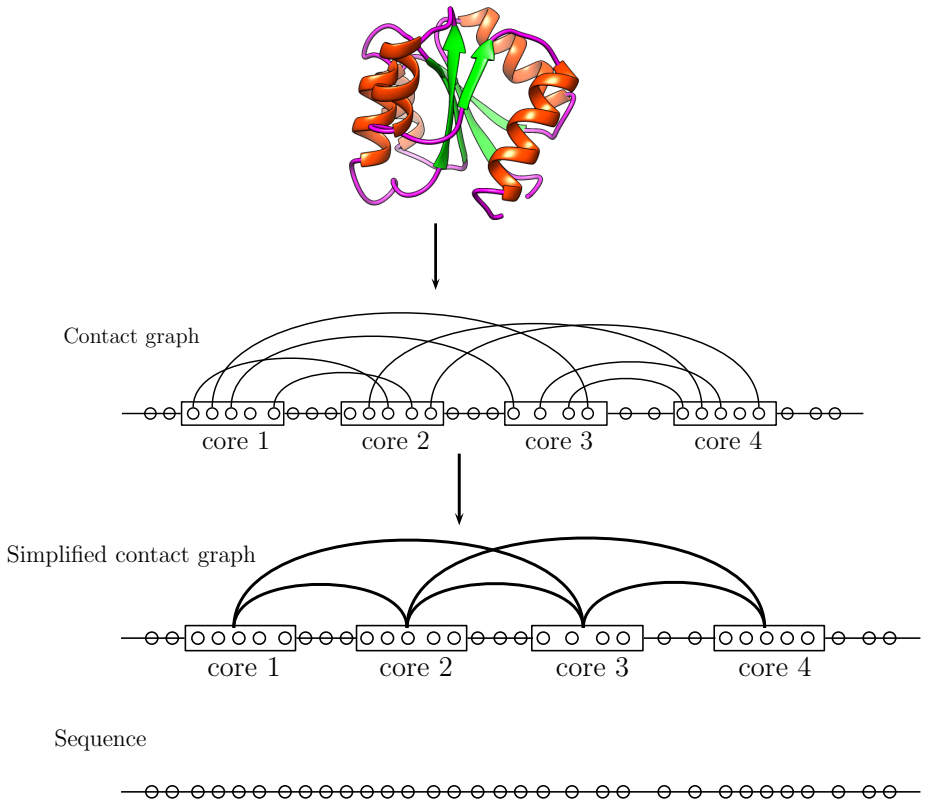


Figure 30: An illustration of the RAPTOR threading algorithm.

Each residue in a template structure is represented as a vertex. One edge connects two residues if they are in contact (their spatial distance is within a given cutoff). The result is a contact graph. Cores are the most conserved segments in the template: Usually alpha-helices and beta-sheets. The contact graph is then simplified such that one edge connects every two cores which are in contact (have at least a pair of contacting amino acids). The query sequence is then aligned to the simplified contact graph. An example of a contact graph is shown in Figure 30.

The problem is formulated as an Integer Linear Programming (ILP) problem as follows:

The variables are:

- $x_{(i,l)}$ denotes core i is aligned to sequence position l
- $y_{(i,l)(j,k)}$ denotes that core i is aligned to position l and core j is aligned to position k at the same time

6.2.2 Alignment as a Linear Programming Problem

Minimize:

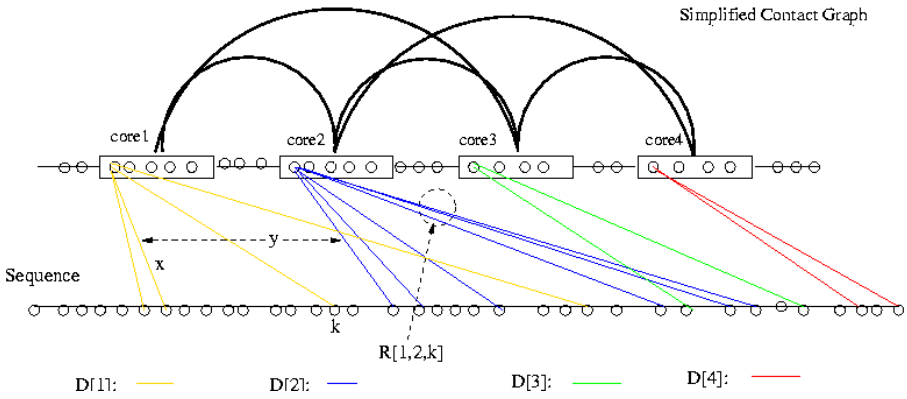


Figure 31: The raptor variables

$$1.E = \sum a_{i,l}x_{i,l} + \sum b_{(i,l),(j,k)}y_{(i,l),(j,k)}$$
$$2.s.t.$$
$$3.x_{i,l}, y_{(i,l),(j,k)} \in \{0,1\}$$
$$4.x_{i,l} + x_{i+1,k} \leq 1 \forall k \in D[i+1] - R[i,i+1,l]$$
$$5.y_{(i,l),(j,k)} = x_{i,l}x_{j,k}$$
$$6. \sum_{l \in D(i)} x_{i,l} = 1$$

E encodes the scoring function $a_{i,l}$ encodes the individual energy terms (E_s,E_{ss},E_m, see above) and $b_{(i,l),(j,k)}$ encodes the pairwise terms (E_g,E_p)

1. The objective function

2. Subject to the following constraints:

3. Binary variables (either put an amino acid in a position or not. Either have two cores aligned to two positions or not.)

4. No conflicts in the structure (no earlier core aligned to a later sequence position)

5. Pairwise and individual terms have to be compatible (not linear but can be made such quite easily)

6. Only one position is selected for each residue

Here is a linearization of condition 5:

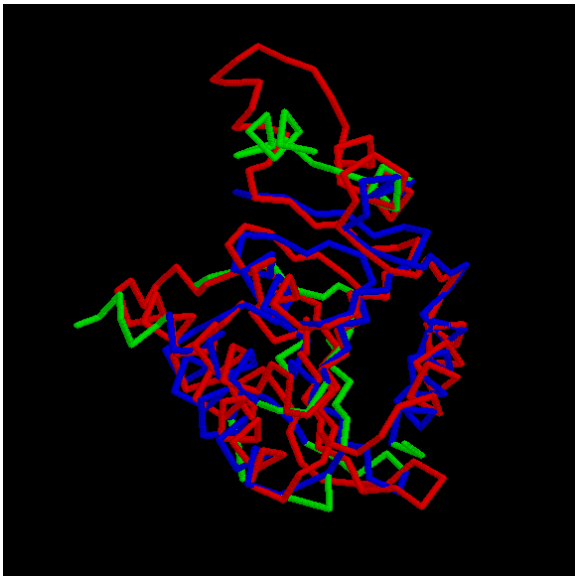


Figure 32: Red: Experimental Structure. Blue/Green: Raptor model. Target Size:144. Superimposable size within 5Å : 118. RMSD:1.9Å

$$x_{i,l} = \sum_{k \in R[i,j,l]} y_{(i,l),(j,k)}, \forall l \in D[i]$$
$$x_{j,k} = \sum_{l \in R[j,k,i]} y_{(i,l),(j,k)}, \forall k \in D[j]$$
$$x_{i,l}, y_{(i,l),(j,k)} \in \{0,1\}$$
$$\sum_{l \in D(i)} x_{i,l} = 1$$

An example of an alignment with RAPTOR is shown in Figure **??**

RaptorX – The next generation:

Integrating global and local context specific information. Includes both alignment and template selection. Currently among state of the art.

Side Chain Placement

An important component of all the above-mentioned protein structure prediction applications is the side chain modeling stage. Side chain modeling is also typically performed as an energy optimization process. In general, the goal of side chain optimization is to identify the most favorable configuration

of the side chains for a given backbone configuration. In many applications, side chain modeling is an independent stage following the backbone structure prediction. While the details may vary, side chain optimization generally involves the following stages:

1. a backbone configuration that provides a structural template
2. a side chain library that provides statistics about likely side chain configuration (this is also called a *rotamer library*
3. a scoring or energy function
4. a search algorithm whose goal is to identify the lowest energy state among all possible configurations.

As you can see, side chain placement can be viewed as a "mini folding" process. Just like the general structure prediction problem, the search space can be continuous or discretized. Searching the continuous space is The use of discrete side chain conformations, called rotamers or conformers, have been very popular.

Currently, the majority of the libraries used for side chain optimization are derivatives of statistical rotamer libraries [**?**]. These libraries are based on the statistical analysis of the distribution of the backbone dihedral angles (the torsional rotations around bonds), which are the main determinants of side chain conformation. The rotamer libraries define the clusters in torsional space, providing their average, distribution and relative population. Following this statistical analysis, it can be seen that for many backbone conformations, only a small percentage of rotamers are statistically probable. The adoption of a rotamer library allows us to focus the search only on the favorable regions of conformational space.

6.2.3 Ab-initio Folding

Ab initio prediction, along with protein design (described next) present the most fundamental, but exciting challenges in computational modeling. Ab initio or *de novo* prediction refers to methods that attempt to predict the tertiary structure based only on the amino acid sequence and the laws of physics. Traditionally, these methods involve prediction of the secondary structures from sequence, followed by conformational sampling and evaluation of tertiary structure to determine the global minimum energy configuration.

6.3 The CASP competition

Critical Assessment of protein Structure Prediction, or CASP, is a community-wide, worldwide experiment for protein structure prediction taking place every two years since 1994 [**?**]. As of 2016, 12 competitions took place. CASP provides research groups with an opportunity to objectively test their structure prediction methods and delivers an independent assessment of the state-of-the-art in protein structure modeling to the research community and software users. In order to ensure no prior information, it is important that the experiment be conducted in a double-blind fashion: The competing group, the organizers and the assessors do not know the structures of the target proteins at the time predictions are made. Targets for structure prediction are either structures soon-to-be solved by X-ray crystallography or NMR spectroscopy, or structures that have just been solved and are kept on hold by the PDB. There is a number of prediction categories: If the given

sequence is found to be related to a protein sequence of a known structure (called a template), comparative protein modeling may be used to predict the tertiary structure. Otherwise, de novo or template-free protein structure prediction must be applied.

The structures are evaluated by comparing the C-α atoms of the target structure to those of the predicted structures. Every model is assigned a numerical score GDT-TS (Global Distance Test–Total Score) describing percentage of well-modeled residues in the model with respect to the target. Free modeling (template-free, or de novo) is also evaluated visually by the assessors, since the numerical scores do not work as well for finding loose resemblances in the most difficult cases.

Evaluation s carried out in the following prediction categories (according to CASP12, `predictioncenter.org`):

- The **High Accuracy Modeling** category includes domains where majority of submitted models are of sufficient accuracy for detailed analysis.
- The **Biological Relevance** category assesses models on the basis of how well they provide answers to biological questions. This category builds on the CASP11 pilot assessment.
- The **Topology** category (formerly Free Modeling) assesses domains where all submitted models are of relatively low accuracy.
- The **Data Assisted** category assesses how much the accuracy of models is improved by the addition of sparse data. Data types are expected to include simulated and actual sparse NMR data, crosslinking data, and low angle X-ray scattering data.
- The **Contact Prediction** category assesses the ability of methods to predict three dimensional contacts in targets structures.
- The **Refinement** category will analyze success in refining models beyond the accuracy obtained in the initial submissions.
- The **Assembly** category assesses how well current methods can determine domain-domain, subunit-subunit, and protein-protein interactions. As in CASP11, CASP works closely with CAPRI, a similar experiment in protein-protein interactions.
- The Accuracy Estimation category assesses the ability to provide useful accuracy estimates for models at the overall, residue, and atomic levels.

Discontinued categories include:

- Secondary structure prediction (discontinued after CASP5 in 2002 due to algorithms reaching high accuracy)
- Domain boundary prediction (CASP6–CASP8)

Starting with CASP7, categories have been redefined to reflect developments in the methods. The 'Template based modeling' category includes all former comparative modeling, homologous fold based models and some analogous fold based models. The 'template free modeling', now 'topology category' includes models of proteins with previously unseen folds and hard analogous fold based models. Since template free targets are not common, in 2011 CASP ROLL was introduced. This continuous (rolling) CASP experiment aims at more rigorous evaluation of template free prediction methods through assessment of a larger number of targets outside of the regular CASP prediction season. CASP results are published in special supplement issues of the scientific journal Proteins, all of which are accessible through the CASP website.

7 Additional Reading

M. Karplus & J. Kuriyan, Molecular Dynamics and protein function, PNAS 102(19):6679-6685 http://www.pnas.org/content/102/19/6679.full.pdf

Energy surface associated with conformational space [Onuchic J.N., Luthey-Schulten Z., and Wolynes P.G. Annu. Rev. Phys. Chem. 48, 1997]

Evolution has "guided" native state (in naturally-occurring proteins) to be lowest free-energy state [Unger R. and Moult, J. Bull. Math. Biol. 55, 1993]

References

[Anfinsen(1973)] Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973. ISSN 0036-8075. URL http://science.sciencemag.org/content/181/4096/223.

[Threading] J. U. Bowie, R. Lüthy, and D. Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 253(5016):164–170, 1991.

[Dill and Chan(1997)] Ken A. Dill and Hue S. Chan. From Levinthal to pathways to funnels. *Nature Structural & Molecular Biology*, 4(1):10–19, 1997.

[Wolynes(1997)] Peter G. Wolynes. Folding funnels and energy landscapes of larger proteins within the capillarity?approximation. *Proceedings of the National Academy of Sciences of the United States of America*, 94(12):6170–6175, 1997.

[Levinthal(1968)] Cyrus Levinthal. Are There Pathways For Protein Folding? *Extrait du Journal de Chimie Physique*, 65, 1968.

[Xu et al.(2003)] Jinbo Xu, Ming Li, Dongsup Kim, and Ying Xu. Raptor: Optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 01(01): 95–117, 2003.