4-26-2007

Constraining Protein Structure Simulation with Aggregate Data Using Robotic Techniques

Eric Bracey
Grand Valley State University

Constraining Protein Structure Simulation with Aggregate Data Using Robotic Techniques

Eric Bracey, Grand Valley State University
Hansye Dulimarta, Ph.D., Advisor
April 26, 2007

Abstract

This approach to protein simulation uses Protein Data Bank information to construct useful, simple, geometric constraints that can be applied to a protein simulation. We compiled experimental data for proteins with between 30–90 residues and analyzed the relationship between their sizes, defined as the radius of a sphere that encloses the 3D structure; the maximum distance between any two residues and the number of residues in the protein. A significant relationship was found and the analysis was used to predict the ranges that the size and maximum distance between residues would have for a protein with a given number of residues. These ranges were used to constrain folding from secondary structures for proteins 1ROP and 1HDD and, using a random path planning approach, produced results that were not terribly accurate, but quite fast, suggesting that the constraint would be most useful as an inexpensive addition to an existing technique.

1. Introduction

Perhaps the most challenging and important problem in molecular biology today is the prediction of protein structure. As the building blocks of all living organisms, proteins are the focus of biochemical research. Understanding their chemistry, their construction and their function is vital to understanding how life works. And the key to understanding their function lies in understanding their structure.

The gold standard for determining protein structure is x-ray crystallography, but this technique is not suitable for all proteins, nor is it always possible to interpret the results. If certain conditions can't be met, and this is not uncommon, x-ray crystallography alone can't tell the observer what the protein looks like. (Branden 1991) One important use of computer simulations of protein structure is to suggest, with a fairly high degree of certainty, what a protein might look like, either by filling in gaps (references) or by suggesting a complete structure.

To date, there have been many attempts, using many different approaches, to find a reliable way to predict the structure of proteins and the way they fold into that structure. While there have been successes, especially with smaller proteins, simulating protein structure has proven to be a difficult problem for several reasons. The primary one is that even though a protein has a structure it is most likely to assume, finding that most likely structure can be prohibitively expensive because of the large number of degrees of freedom involved. Theoretically, it is possible to compute the structure of a protein given only its amino acid sequence, but, because of the huge number of possible configurations,

calculating that structure for larger proteins would take until the end of time given current methods and technology.

For this reason, much of the research on the subject has tried to find ways to shorten the time needed to do a simulation. Many methods have been tried and some of those will be discussed below, but one of the more ingenious has been the application of robotic motion planning techniques to the problem of protein formation. (Amato, Dill et al. 2003)

At first glance, the connection between the two is not obvious, but the key lies in viewing a protein as a chain of amino acids (figure here). In robotics, one of the most studied problems is how to move an articulated robot arm. An arm can be viewed as a chain, with one or more degrees of freedom at each joint, and so can a protein. By making this connection, it is possible to apply some of the concepts and algorithms of robotics to protein structure simulation. (references) Since robot arms are designed to move in real time, there has been a considerable amount of effort devoted to inventing faster algorithms for robot motion planning (references to robot motion planning algorigthms) and this research has been productively applied to proteins. (Gupta 1998)

One kind of motion planning algorithm, known as random path planning, takes advantage of the likelihood that there is more than one path from one configuration to another. (reference) According to recent research, proteins likely have the same property, (reference) so the algorithm has been applied productively to the study of protein folding.

(reference) The disadvantage of random path planning as applied to proteins is that the configuration space is based on two known configuration, the protein's fully extended state and its fully folded state. A useful extension would be an alternative method of creating configurations, which is what the experiment described in this paper attempts to do, by using the information compiled on many proteins to make some predictions about the expected size of a protein, which is used as a basis for creating a configuration space.

In this paper, we will focus primarily on the reasoning and the techniques involved in creating such a configuration space as a means of exploring the idea's feasibility. Possible applications, which will be discussed in greater detail in the Future Directions section of this paper, could include using the technique as a filter, limiting the configuration space another technique works with, (reference) or as the basis for the creation of an ab initio configuration space. The rest of this paper will give some background on proteins, robotic techniques and their intersection, in order to provide a context for the actual experiment.

2. Basics of Proteins

2.1. The Central Dogma

The process of building a protein starts in the cell's DNA. While exploring all the complexities of DNA and the translation of its information into a protein is beyond the scope of this paper, a brief, and simplified, summary of the process is in order.

An organism's genome is the complete set of DNA that encodes all the information needed to build the proteins that constitute it. DNA itself is composed of two connected strands of nitrogenous bases, linked by a sugar-phosphate backbone. The four bases are Adenine (A), Thymine (T), Cytosine (C) and Guanine (G). The two strands mirror each other in a predetermined fashion, with A always linked to T and C always linked to G. The bases are grouped into triplets, known as codons, which can potentially represent an amino acid, a protein's starting point or a protein's ending point. Though there are sixty-four possible combinations of codons (43), they only represent twenty-five distinct entities. Of these, twenty-two are amino acids, though two of these are extremely rare; two are stop codons; and one is the start codon. Often, only the first two elements of the codon are sufficient to encode the amino acid and the third element of the codon is redundant. (Bergeron 2003)

A protein is derived from a sequence of codons in the DNA which are delimited by start and stop codons. When a protein needs to be formed, the appropriate DNA is chemically split and one of the strands is replicated as messenger RNA (mRNA). The messenger RNA is passed through the nucleus membrane and into the cytoplasm where the particular sequence needed to produce the desired protein is edited out. Once the correct sequence has been assembled, it is fed into a ribosome, which is a cone-shaped biological factory that creates proteins. Beginning from the start codon, the codons tell the ribosome which amino acids to form and in what order, continuing to the stop codon. As

the amino acids are produced, they emerge, linked together in a long chain, from the ribosome.

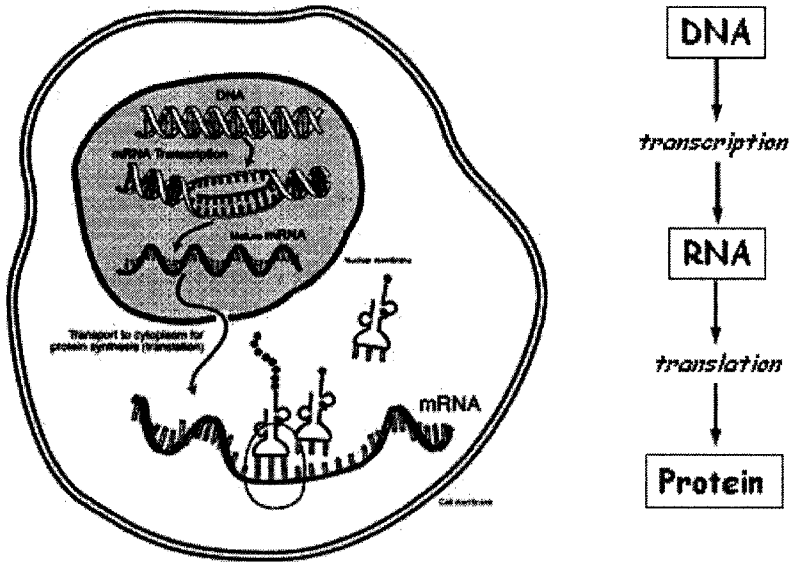


Figure 1. DNA transcription and transport

2.2. Basic Chemistry of Proteins

As mentioned before, a protein is a long chain of amino acids linked together. Structurally and chemically, each amino acid is built around a central carbon atom

(Calpha) which is linked to four distinct elements: an amino group, a carboxyl group, a.

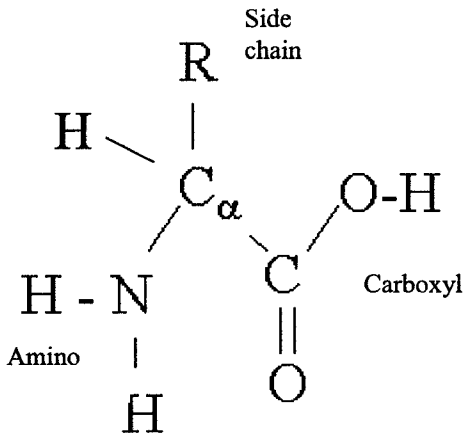


Figure 2. Generic Amino Acid Structure

lone hydrogen atom and a side chain. Only the side chains differ amongst the twenty common amino acids, the other elements are common to all.

The amino acids are linked end-to-end by peptide bonds, which are formed when the amino group of one amino acid combines with the carboxyl group of the one following it, linking the nitrogen atom of the amino group with the carbon of the carboxyl group and shedding a water molecule.

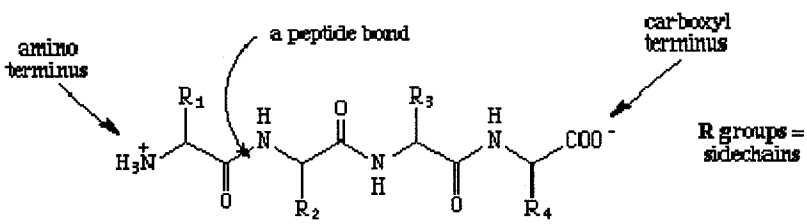


Figure 3. Protein chain

This means that the protein will always start with a free amino group at the start and a free carboxyl group at the end. (Clote 2000) Each link in the chain, therefore, is made

up of the central carbon atom, which is still linked to the side chain and hydrogen atom it began with, flanked by one (at the ends of the chain) or two peptide bonds. Links in the chain are commonly called residues. Other types of chemical bonds can develop between the residues in a protein, and can influence or stabilize its final structure. Proteins found in nature can range from as few as ten residues (though proteins that have fewer than 20 residues are often referred to as peptides and do not form complex 3-D structures) to as many as a 1000, but tend to average around 200 – 350 residues.

The peptide bonds between the residues are nearly rigid, but the connections between Calpha and the peptide bonds flanking it are not. On one side, Calpha is connected to the nitrogen atom remaining from its amino group and on the other to the carbon atom (C') remaining from its carboxyl group. Therefore, each residue has two dihedral degrees of freedom at these points which give the protein the flexibility to fold. The angle of rotation around the N-Calpha bond is commonly called phi (Φ) and that around the Calpha-C' bond is commonly called psi (Ψ). Generally, these are assumed to rotate around the Z-axis. The range of motion for these two angles is limited by possible collisions between the side chains and the main chain. These physical limits are commonly referred to as steric limits and the collisions are referred to as steric collisions. It is possible to plot what combinations of angles are possible for phi and psi depending on the composition of the side chains. This diagram is called a Ramachandran plot after its originator. (Ramachandran 1968)

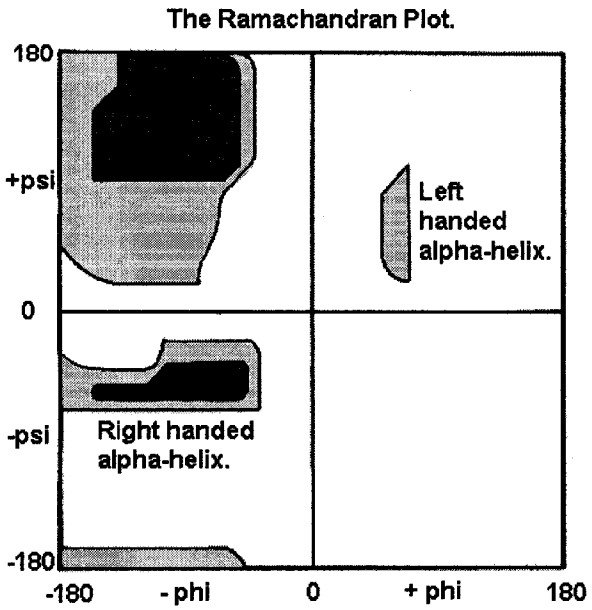


Figure 4. Ramachandran plot showing secondary structure correlation

For most residues, the same combinations of angles are possible, and these substantially limit the range of motion. The chief exception is the amino acid Glycine which has only an hydrogen atom for a side chain, giving it a much wider range of motion without steric clashes because of its small size. (Branden 1991)

Steric collisions are not the only important constraint on a protein's possible configuration. Most residues have side chains that are hydrophobic, repelled by water, or hydrophilic, attracted by water, and this has a major influence on how a protein is structured. In general, a protein folds so that its hydrophobic side chains are clustered in the core of the protein, away from the water that permeates the cell, and with its hydrophilic side chains on the outside of the protein.

2.3. The Folding Process

As soon as the protein begins to emerge from the ribosome, it begins the folding process, so the protein begins assuming its final shape even before the ribosome finishes producing it. There are many different types of proteins, and great variation in structure between them. As a protein folds, the hydrophobic and hydrophilic side chains interact with the watery medium in the cell and begin pushing the protein into the shape it will assume. Steric collisions and intra-molecular forces limit the protein's range of motion and push or pull its residues apart. In some cases, helper molecules interact with the protein chain and further influence the folding process. Given all this complexity, how can any system, any pattern, be determined for the folding process?

As mentioned above, there is no way to accurately compute what a protein's structure should be based only on its amino acid sequence. Molecular biologists have used x-ray crystallography to determine what the actual structure of proteins is. While this hasn't yielded a general, over-arching rule for protein formation, it has given biologists enough information to talk about the folding process as having three or four stages, depending on the protein. These are called, naturally enough, the primary, secondary, tertiary and quaternary structures. On its journey to its final shape, the protein passes through these structures, which are intermediate steps on its way to its final, stable form

The primary structure is a line consisting of the long chain of amino acids and is more theoretical than actual, since folding begins as soon as the protein starts forming. The

secondary structure forms largely as the result of the interaction of hydrophobic and hydrophilic side chains. The main chain of a protein, the backbone, is hydrophilic and so resists being folded into the core away from water. Because of this, an hydrophobic side chain and the hydropholic main chain it is attached to pull in opposite directions, making it difficult for the protein to stably fold. The secondary structure of a protein deals with this by folding in such a way that the hydrophilic properties of the main chain are neutralized by hydrogen bonds between residues on the chain. (Branden 1991)

In general, this means that local areas of the protein from into secondary structures of one of two types: alpha helices or beta sheets. These are regular structures which occur when the forces at play in the protein force phi and psi angles to be the same, causing the creation of hydrogen bonds that stabilize the structure into a helice or sheet shape. Most proteins consist of combinations of the two types, though the beta sheet formation is more common. The secondary structures are connected to each other by portions of the amino acid chain of the protein that have not coalesced into helice or sheet forms.

Once the protein has settled into its secondary structure, it folds into its tertiary structure. The tertiary structure arranges the secondary structural elements, usually alpha-helices and beta-sheets into a final shape, arranging regions that didn't fall into a secondary structure appropriately, generally into different loop configurations that arrange the hydrophobic sides of the secondary structure so that they face into the core of the protein, away from the watery medium of the organism. For proteins that consist of only one amino acid chain, this is the final step in the folding process.

For proteins consisting of more than one amino acid chain, there is one more step in the folding process: quaternary structure. These proteins are especially idiosyncratic and are often referred to as protein complexes to differentiate them from proteins that consist of a single chain. The simulation techniques discussed in this paper do not attempt to describe quaternary structure.

As mentioned in the introduction, in the course of folding, a protein can potentially take many paths to its native state, or minimum energy configuration. (reference) This configuration is defined as the one in which the protein has the least possible free energy and is at its most stable. As shown in the figure below, the transition from an unfolded to folded state is not usually a smooth one, as its possible for the protein to become temporarily stuck in local minima before reaching the native state. As the protein nears its native state, the number of potential pathways decreases and local minima may become harder to recognize. In reality, some proteins may not have one stable low energy configuration, but several, though simulations assume only one.

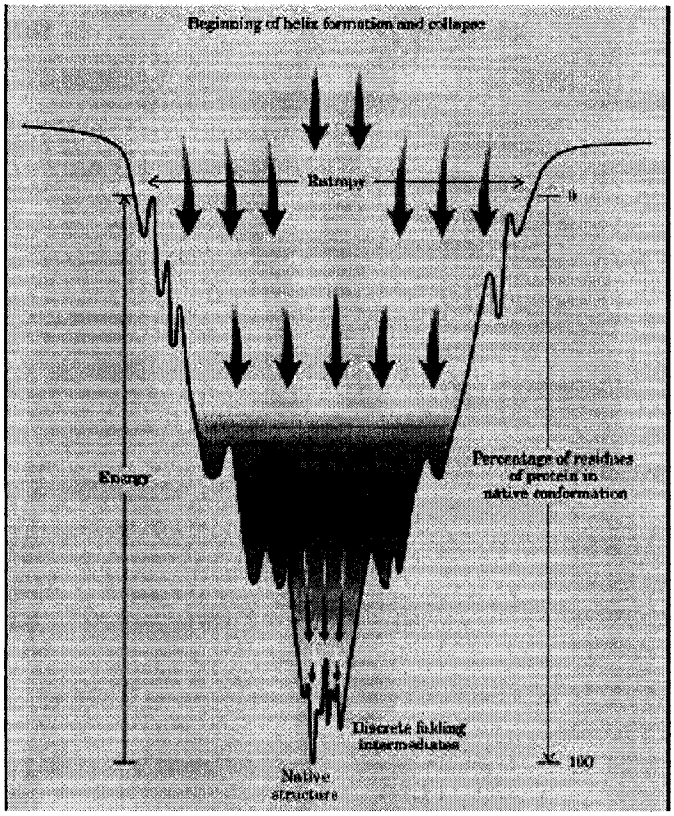


Figure 5. Free energy landscape

2.4. Protein Simulation

Since proteins can be x-rayed, why is finding a way to simulate their structure so important? The primary reason, as briefly mentioned in the introduction, is that x-ray crystallography has serious limitations. Not all proteins can be crystallized, especially membrane proteins, and making a crystal is a painstaking process that doesn't always produce usable crystals. Even if one gets an usable crystal, the medium of the crystal is very different from the medium the protein works in, so there are likely differences between the measured configuration and the real world configuration. But the biggest

obstacle is that not all the results from doing x-ray crystallography of protein crystals can be interpreted, because of the phase angle problem. (Branden 1991) An x-ray crystallography is taken by sending an x-ray stream through the crystal which diffracts and creates a pattern on the film or electronic device. Since the results are saved in a 2-D form, an interpreter needs to know enough information to translate them back into a 3-D form. Unfortunately, the film only shows the amplitude of the wave that created it. The wavelength is known because this is controlled by the experimenter, but the phase of the wave, which is the last piece of information needed to figure out where the atom represented by a spot on the film is, is not.

In crystallography, there are a couple of methods used to work around this problem. These are molecular replacement and heavy metal soaking, which involve comparisons to known structures and trying to insert heavy metal atoms into the crystal, respectively. Unfortunately, these methods do not work in all cases and, even if a result is obtained, not all results are reliable. Many variables, including the quality of the crystal, can cause them to be inaccurate. One way to double-check results or to provide a basis for interpreting what a protein x-ray shows is to do a computer simulation.

Given infinite time, a computer simulation could examine all possible configurations of a protein and discover its preferred or native configuration. Since no one has infinite time, the big challenge in computer simulation of protein structure is to find a way to produce reasonably accurate simulations in a reasonable amount of time. The obvious way to do this is to try to find ways of limiting the number of configurations you need to examine

while doing your simulation. Unfortunately, the simplifications and assumptions needed to bring the computation time down to a reasonable level force a tradeoff between time and the quality of the simulation.

As you look at larger proteins, and the average protein is considered quite large when doing simulations, the computation time becomes an insurmountable obstacle, though large advances have been made over the years. Even so, existing simulation techniques have proven useful, especially in conjunction with x-ray crystallography, in providing a realistic starting point for the interpretation of crystallographic data. For smaller proteins, simulations are capable of generating precise models. Most simulation approaches fall into two broad categories, comparative and structural. This paper's approach is a synthesis of the two, so both will be discussed.

2.4.1. Comparative Approaches to Simulation

Comparative approaches use the existing data on protein structure gained from prior x-ray crystallography. The genetic sequence or amino acid sequence of a protein of unknown structure is compared to the sequences of proteins with known structures in an attempt to find the closest fit. Where the sequences match, it is probable that their structures will also match. Obviously, sequences that exactly match are the best indicator, but near matches are also likely to have similar structures since protein structure and function are often not affected by the substitution for one amino acid for another in a protein's sequence. Such substitutions are common over time, due to mutations, and biologists have determined the likelihood that one amino acid will

mutate to another. This information is usually presented in grid form, like the Blosum substitution matrix, which lists the twenty common amino acids on each axis and lists the probability that one will mutate to another. (Bergeron 2003) This kind of data is a useful tool for refining possible matches. Often, proteins with small variations in their amino acid sequences due to mutations will maintain the same shape as their predecessors, so this can be an effective way of predicting the novel structure. Protein threading, which tries to match both the sequence and the shape simultaneously, is an interesting example of this approach. (Clote 2000)

In the absence of strongly similar proteins, however, the comparative method gives poor results. The method can instead focus on trying to resolve secondary structure by identifying the portions of the structure most likely to resolve themselves into alpha-helices, beta-sheets or combinations. This has met with some success, but a high success rate for secondary structure predictions is still only in the 75% range, (Rost 2001) which leaves plenty of room for improvement.

2.4.2 Structural Approaches to Simulation

Purely structural approaches try to determine what a protein's structure is based only on its amino acid sequence, or an ab initio approach. The basic assumption behind these approaches, which has not yet been proven, is that there exists a configuration for the protein that results in the lowest possible total energy to maintain it, and this configuration is the preferred or native configuration for that protein. The total energy is viewed as a combination of the different forces at work in the protein, including Van der

Waals forces, the hydrophobic factor, electrostatic force between charged amino acids and the Leonard-Jones potential. Currently, there is no accepted standard energy function and several different approximations have been used.

A key for many energy functions is the Boltzmann distribution. This was originally devised in the late nineteenth century by L. Boltzmann to describe the energy level of a volume of molecules in an ideal gas at a certain temperature. It has been discovered by compiling tables of relationships between amino acid residues in known structures that the distance and, presumably, the energy, between pairs also follows a Boltzmann distribution. (Clote 2000)

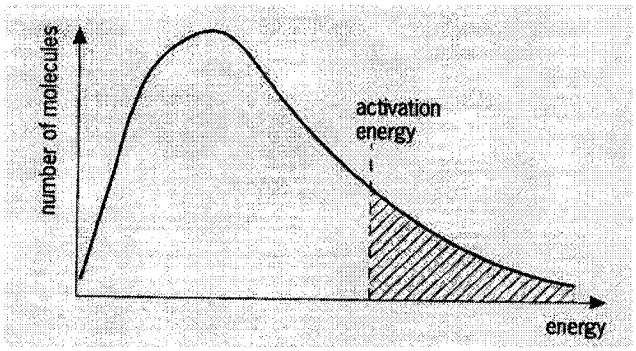


Figure 6. Boltzmann distribution for gases

Therefore, an energy function for a protein can be derived by comparing the Euclidean distances between proteins to the experimentally observed distances to get the probability that a given configuration is near the energy minimum. By summing these values for all pairs of contiguous amino acids, you get the total energy for the entire configuration. (Sippl 1990) One of the interesting characteristics of this distribution is that it is based on temperature. At low positive temperatures the distribution assumes the shape shown in the diagram above; at high temperatures it approaches a normal distribution.

With an energy function based on distance between residues, it is now possible to simulate a protein from ab initio. By changing a configuration randomly a few residues at a time and retesting the energy function, it is theoretically possible to eventually find the lowest energy configuration, though special provisions have to be made to avoid local minima. This is commonly referred to as the Monte Carlo technique and many refinements and variations exist. Simulated annealing is one variant of Monte Carlo simulation that adds the technique of changing the simulated temperature in order to cause greater perturbations by altering the probabilities of the Boltzmann distribution, which changes shape as the temperature increases. The expectation is that greater variations will give a greater chance of avoiding local minima, but this is not guaranteed. This process is designed to shake the simulated protein out of any local minima by simulating its heating and cooling, causing it to unfold and refold. (Kirkpatrick 1983) Unfortunately, these approaches are NP-hard, i.e. there is no guarantee of finding a solution without searching the entire problem space, which is practically unsolvable with a protein of any size. (Clote 2000)

A common method of representing a protein is to abstract it as a 2D or 3D lattice with the following assumptions:

- All bond lengths are equal.
- All residues are the same size.
- All residues must be located at points in the lattice. (Clote 2000)

Sippl's energy model, based on Euclidean distance, doesn't work well with the assumption that all bond lengths are equal, so other energy models are used in a lattice model. One example is the HP model, which simplifies the problem by focusing on protein's hydrophobic properties. The energy between pairs is represented by a simple matrix, where hydrophobic residues are represented by H and other residues are represented by P:

r1 \ r2	H	P
H	-1	0
P	0	0

Hydrophobic residues that are adjacent to each other in the lattice, whether they are connected or not, contribute -1 to the total energy, all other combinations contribute 0. The total energy is determined by comparing all adjacent pairs in the matrix. As with Sippl's method, random moves that lower the total energy are accepted. (Lau 1989) Again, local minimas are a problem and the technique has also proved to be NP hard.

3. Robot Path Planning

Before discussing protein simulation further, it is important to explore some of the basic premises of robot path planning, so that the intersection between the two can be clearly understood. One of the basic problems in robotics is navigation. The problem of how to get a robot from point A to point B, or, for a robot fixed at one end, like a robotic arm,

how to get it from configuration A to configuration B, has been approached from several angles. Robots that physically change location and move through changing or unknown environments generally rely on sensors. Their algorithms tend to be a mix of reactive techniques, which are simple behaviors activated by a stimulus, and some sort of path-planning, using maps, landmarks or some combination. In the case where a robot functions in a completely known environment, which is rare in the real world but a useful assumption for simulations, the preferred navigation method uses some kind of path planning technique.

3.1. Collision Detection

The most basic task involved in trying to figure out how to move a robot is detecting when the robot collides with itself or something else. Without an efficient means of testing for possible collisions, possible paths can take too long to compute. Touch sensors can send a signal when the robot physically hits something, and a reactive algorithm could react appropriately, as in the case where a toy car runs into a wall and spins to face in a different, randomly chosen, direction before moving forward again. In most situations, including simulations, different techniques must be used both because not all situations can rely on sensors and because the preferred goal is to avoid collisions rather than dealing with them after they occur.

In situations where all the objects involved and their dimensions are known, the Euclidean distance between different parts of the robot and different obstacles can be computed. Unfortunately, if the robot moves, these distances all need to be computed again, which becomes computationally expensive. (Mirtich) One way of limiting the amount of work done at each step, is to take advantage of the notion of coherence, which is the realization that the relationship between most of the objects in a scene and each other doesn't change much every time something moves, so only the changes need to be considered. To do this, a scene must have some kind of memory.

One way to describe where things are in a scene is by using bounding volumes to describe the objects involved. A bounding volume is a geometric shape that completely encloses an object. It can then be recursively broken down to more precisely describe where its elements are in space and stored as a tree.

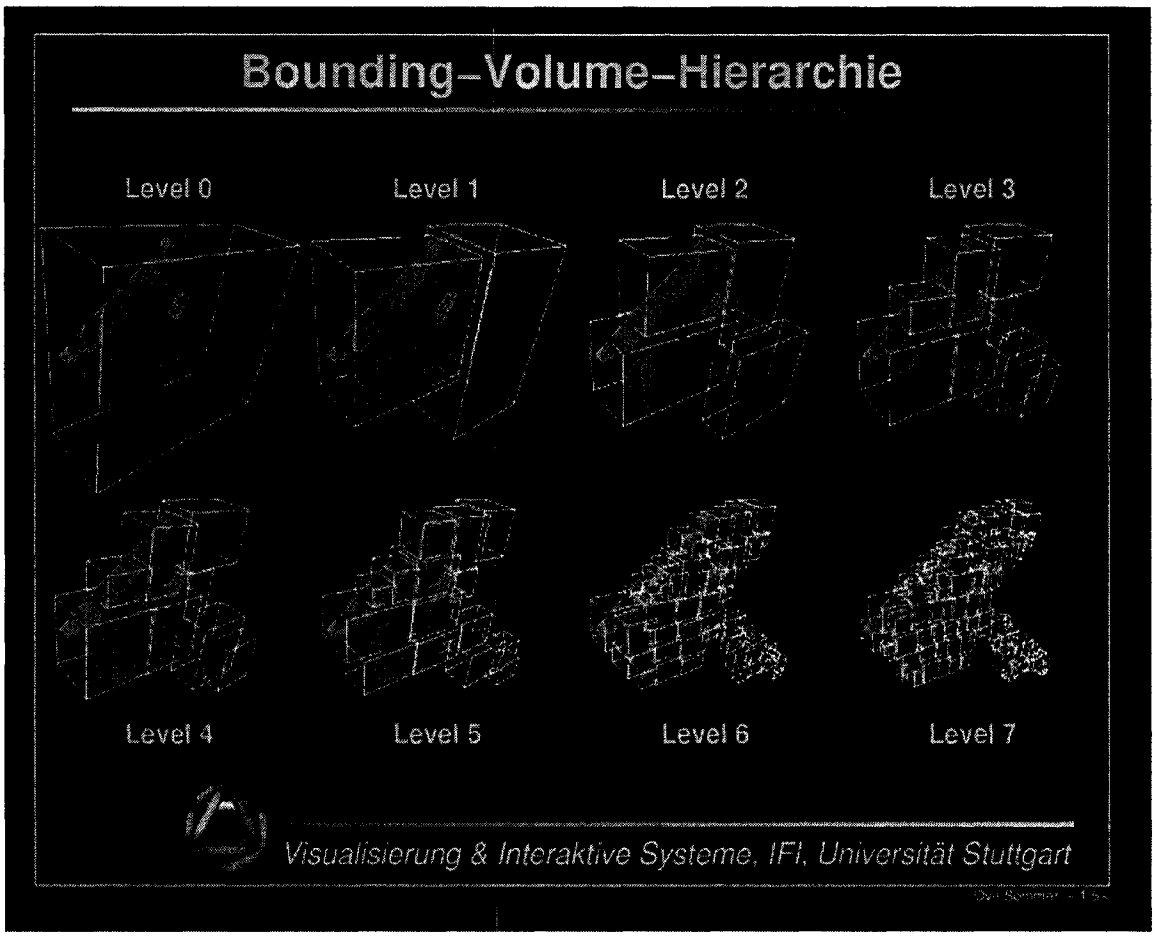


Figure 5. Hierarchical bounding volume example

When checking for collision between two objects, their trees can be compared to see if any of their top-level objects intersect. If so, the next level is compared, and so on, in a binary search of the tree until the bottom level is reached and a precise determination (or as precise as the tree's resolution will allow) can be made. The tree can be saved as a linked list or as a set of hash tables, with each table representing on level of the tree, a technique known as hierarchical spatial hashing. (Murphy 2000) This kind of representation generally treats all objects as concave.

Other techniques for collision detection include coordinate sorting, where the coordinates of bounding volumes are maintained in a sorted list. The volumes overlap if and only if their coordinates overlap in all dimensions. The Lin-Canny closest features algorithm is used when precise measurements of distances between objects are needed. It creates geometric volumes between shapes by extending lines from their vertices to make Voronoi regions. If the regions of two vertices intersect, then they are oriented towards each other and can be tested for closeness between the edges, vertices and planes of the objects. (Murphy 2000)

3.2. Configuration Space

In order to do path planning or many reactive techniques, you need some sort of abstract representation of the space that the robot exists in, or a map. A basic map shows the limits of the space and the obstacles contained in it. The representation should accurately show the location and orientation of the robot and any obstacles, and allow the robot to assume any configuration possible for it. This map and its corresponding data structure are often referred to as the configuration space or Cspace of a problem. A Cspace representation is created by using some method to abstract the actual space into a useful data structure. Some methods of breaking up the space are regular grids, Voronoi diagrams (which compute a line equidistant from all obstacles), or bounding trees (which are trees describing the possible obstacles recursively using progressively smaller bounding volumes). (Murphy 2000)

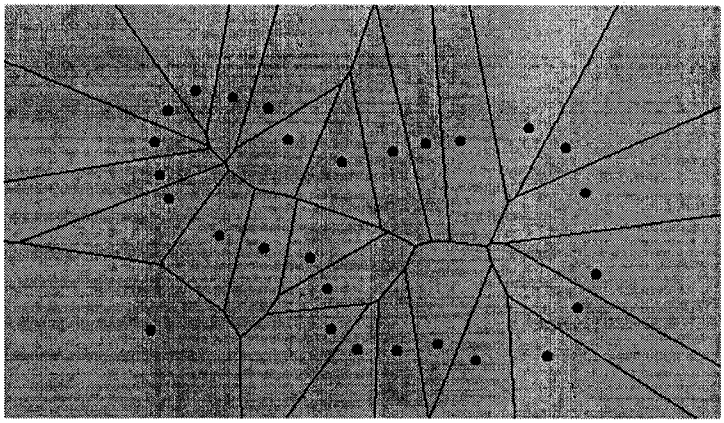


Figure 8. A Sample Voronoi Diagram

The Cspace representation is often also used to condense the degrees of freedom of the robot so that the problem can be represented in 2D or 3D space. Robots that move from one point to another in 3D space actually have up to six degrees of freedom. They can move in x or y direction, they have a height, z, and three more degrees, pitch, yaw, and roll, are needed to represent the robot's facing and tilt with respect to the plane it moves on. In most cases, a robot of this type can be treated as having only three degrees of freedom (x, y and pitch) if they are assumed to be able to turn in place (holonomic), so that facing is not an issue. This is a reasonable approximation for some robots, but a robotic arm has a degree of freedom at each joint which is intrinsic to its function, and so cannot be abstracted away. This means that the Cspace for a robot with multiple degrees of freedom can rapidly become very large. A robot arm with six joints, or six degrees of freedom, each having a range of motion of 90 degrees would, if you broke the range up into one degree increments, have 90^6, or more than 500 billion configurations. [Challou in Gupta] So how you construct a Cspace is an important part of the problem.

Once a Cspace has been created, it can be used to determine how a robot will go from one configuration to another. An algorithm for this purpose can be resolution complete, which means that it will find a path if one exists, or probabilistically complete, which means that it approaches a probability of one for finding the path if one exists. Probabilistically complete algorithms approximate a solution in order to arrive at a conclusion more quickly. The tradeoff is that a higher degree of certainty requires more computing time. . [Gupta 4]

The popular Cspace representations lend themselves well to treatment as a graph and the algorithms used to compute a path take advantage of this. The classic algorithm is known as A search and incrementally calculates a path from a start node to a goal node by using a function to calculate which possible node gets it closest to its goal. [Murphy 361] A complete version of this examines all possible paths from start to finish. Another version, known as A*, instead creates an ideal path, ignoring obstacles and computes the next best node by its closeness to the ideal path. These methods can be computationally expensive and break down when a robot has more than six degrees of freedom.

A popular reactive technique that uses a Cspace representation is the potential field algorithm. This models the space as an array of vectors whose magnitude and direction function as a kind of force field, pushing the robot away from obstacles and toward the goal state. Obstacles will be surrounded by vectors that tend to push the robot away from them and the rest of the space will have vectors pointing towards the goal state. If there

is no specific goal, the technique is still useful as a way of controlling where in the space the robot may move.

3.3. Probabilistic Roadmaps

A technique designed to deal with robots with high degrees of freedom is the probabilistic roadmap planner approach of Kavraki and Latombe. (Kavraki 1998) As above, the Cspace must be defined in some way, but instead of trying to find different, specific, collision-free paths in a linear fashion, the goal is to create a space where many possible paths may exist. The key insight of the probabilistic roadmap is that there may be multiple collision-free paths between a start state and a goal state.

The first phase of the process generates a subset of Cspace which is called Cfree. Cfree is the configuration space where the robot can exist without colliding with itself or other obstacles. Of course, to completely define Cfree would be too expensive, so the first phase consists of selecting possible configurations at random within the Cspace and checking that they are collision free before adding them to Cfree, which is stored as a data structure, or file if the configuration space is extremely large, to be referenced by the next phase of the planner. The configurations in Cfree are further categorized by determining which configurations can be reached directly by other configurations. This is done by a simple local planner which can use any path-planning technique, such as linear distance, potential fields, etc, but the best results are obtained if it is fast, simple

and deterministic. The resulting map is a directed graph, where each configuration is a node, connected by edges to other configurations it is possible to move to from it.

In its simplest form, the next phase tries to create a path that answers a specific query, where the query is a request to find a path that connects a start configuration (Q) to a goal configuration (Q'). For any query that is possible, if the Cfree configuration has sufficient coverage, the probability that the local planner will find a path should approach one. The classic algorithm for computing the path is to start at both Q and Q' and move in a stepwise fashion through the map and towards each other until the two paths are linked. If the link is difficult, the process will probably have to backtrack in order to find it. For extremely difficult paths, the initial map creation may not have found them or it may take an extremely long time to find them. For this reason, an implementation of a probabilistic roadmap will probably have some kind of backtracking limit (either time or number of iterations), which, when reached, will signal that a path can not be found.

A useful refinement of the process is to add an intermediate phase where the roadmap's graph is examined for regions where few or no nodes connect to neighboring nodes. It can be assumed that such regions are particularly difficult or impossible to navigate, so adding additional nodes in them will make it easier for a query to be answered. This is done by creating additional configurations (or nodes) in difficult regions and adding them to Cfree. (reference)

4. Protein Structure and Robotic Path Planning Techniques

The key insight that needs to be made in order to apply robotic techniques to simulating protein structure is the understanding that a protein is, physically, a chain of amino acid residues. Each link of the chain has two degrees of freedom, described earlier as the psi and phi angles, whose range of motion is constrained by the limits described in the Ramachandran diagram (fig. 4). If some of the simplifying assumptions of the lattice method are also applied, like assuming that all bond lengths are equal, a protein starts to look a lot like a robot arm.

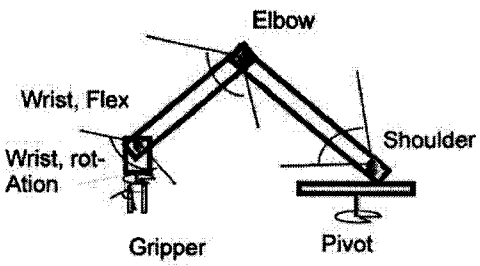


Figure 6. A Simple Robot Arm

The analogy extends further than the idea of a protein as a chain, however. There are similarities between techniques used to simulate proteins and some of the techniques used in robot path planning.

One of the more common techniques for protein simulation, Monte Carlo simulation, can be compared to the potential field technique for finding a path from one point to another. In the potential field technique, the configuration space is constructed of vectors that all point towards the goal state, in a sort of primitive energy function. If a move takes the

robot closer to that state, as computed by summing the vectors of that region, it is accepted; otherwise, it is not. In Monte Carlo simulation, the lowest energy configuration is the desired goal and every random move that reduces the protein's energy is accepted, while moves that increase the energy are generally rejected. (reference) For both potential field and Monte Carlo simulations, the chance of being trapped in local energy minima is very high, so they allow choosing sub-optimal moves with a certain probability. (Apaydin, Brutlag et al. 2003)

One of the more promising applications of robotics ideas is the use of the Probabilistic Roadmap technique to protein simulation by Amato and Song, (Amato and Song 2002) which is related to earlier work on ligand binding using the same technique. (Singh 1999) Their approach makes some of the simplifying assumptions discussed above, such as assuming that all bond lengths are equal, and that the psi and phi angles are the only flexor points in each residue. The major difference between using this approach with robotic motion planning and proteins is that, for robots, the configuration must only be collision-free; and for proteins, the protein seeks its lowest energy, or natural, state. Therefore, when constructing the roadmap for a protein simulation, nodes in the graph are linked together with edges weighted by a probability derived from an energy function that one node, or state, will change to the configuration of another, rather than being unweighted as in the classic Probabilistic Roadmap technique. By randomly creating configurations throughout the possible configuration space for the protein, it becomes much easier to avoid the local minima and maxima that can cause problems when using Monte Carlo methods.

In Amato and Song's initial work, modeling was done using known protein structures as a starting point for the simulation, as a way to explore how folding works, but other later papers (Amato, Dill et al. 2003) have explored ab initio folding. The random path planning technique makes it quicker to test many possible foldings by picking representative configurations randomly across the sample space. While the technique has shown some promise in the closeness of its simulations to known proteins, the authors have not published results for proteins greater than 60 residues in size in either paper.

In another recent development, Apaydin, et al have developed another application of the Probabilistic Roadmap technique to protein simulation, calling it the stochastic conformational roadmap. (Apaydin, Guestrin et al. 2002; Apaydin, Brutlag et al. 2003) This technique doesn't try to simulate all possible individual configurations of the protein in detail, but instead tries to generate a field of possible configurations expressed as probabilities, according to a Markov Chain model. This technique takes advantage of some of the ensemble properties of molecular motion and the belief among some biologists that proteins can pass through a number of possible configurations before reaching their natural state. By representing the possible configurations as a Markov Chain, possible configurations are represented as energy values and assigned probabilities, rather than being some kind of 3D representation. The technique is not as precise as some others, but its results tend to follow the Boltzmann distribution in the limit, just as some of the other simulation techniques, like Monte Carlo, do.

A recent paper by Bedem, et al, uses an inverse-kinematics approach to fill in gaps in protein structure in the case where portions of a structure have been determined by some other method. Using the sections that have been determined, the algorithm tries to figure out what the unknown loops might look like. Like roadmap planning, this technique generates a set of possible configurations randomly according to the expected distribution over the available space and selects the best fitting with a local planner.(van den Bedem 2005)

5. Explanation - Using data to constrain structural searches

Any approach to simulation must, because of the enormous number of degrees of freedom of the typical protein, be an attempt to somehow limit the number of possible protein configurations being examined during the simulation. A random path planning approach is a reasonable way to approach this because it samples the configuration space randomly, under the assumption that there are multiple paths from one configuration to another.

Almost all the papers that use this technique, however, base their randomly sampled configurations on two non-random configurations, the native state of the protein and its fully extended configuration. The new configurations are created by randomly varying one or more of the degrees of freedom in the original configuration, assuming that these values are normally distributed, and testing the new configuration for an acceptable energy level before adding it to the configuration space. (some references here)

While this approach has yielded interesting results, especially in looking at how proteins fold, the configuration space it uses is still quite large. How, then, can the configuration space be reasonably constrained?

The online Protein Data Bank (rcsb.org/pdb) (reference) contains a great deal of information about protein configurations extracted from the experimental literature. This information is often used for comparative (homogeneous) approaches to protein simulation, but rarely for structural approaches, though some variables like bond length are commonly based on estimates derived from experimental data. An example of an approach that has both structural and comparative components is the work of Ngan (Ngan, et al 2006) and Samudrala and Levitt (2002), which looks at the physical conformations of protein doubles or triplets in many proteins in order to create scoring functions used to build up a protein from fragments when simulating new configurations geometrically.

This suggests that one way to constrain the configuration space for a path planning approach is by using experimental data to create either physical limits or probabilities for ranges of motion. One gross approach, and the one used here, is to compile experimental data for a set of proteins and look at the relationship between their sizes, defined as the radius of a sphere that encloses all the 3D coordinates of a protein structure;b the maximum distance between any two residues and the number of residues in the protein. If there is a significant relationship, then the size and the maximum distance between points, or at least their likely range, can be predicted and used to constrain a simulation.

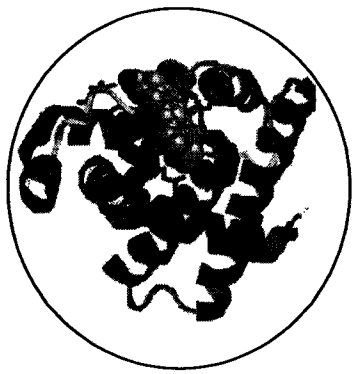


Figure n. Example of a bounding sphere

6. Analysis of Experimental Data

From the Protein Data Bank, we pulled all single-stranded proteins which had between 30 and 90 residues, the range typically used when looking at new protein simulation techniques, giving us about 2200 proteins. A simple Java program sorted through the pdb files and extracted the following pieces of information for each protein:

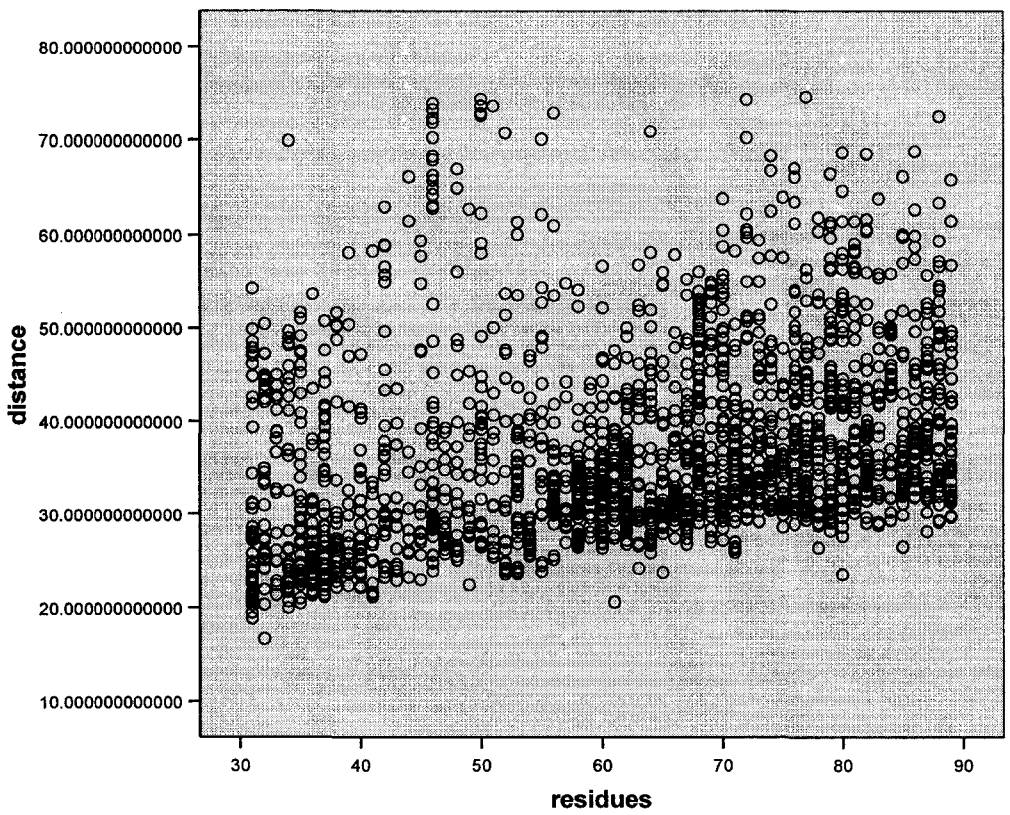
- Bounding sphere radius
- Maximum distance between residues
- Residue furthest from center of bounding sphere in a loop secondary structure (Boolean)
- Residue next furthest from center in a loop (Boolean)

The bounding sphere radius is based on the difference between the mean of all the Calpha coordinates in the protein configuration and the point furthest from that mean. The

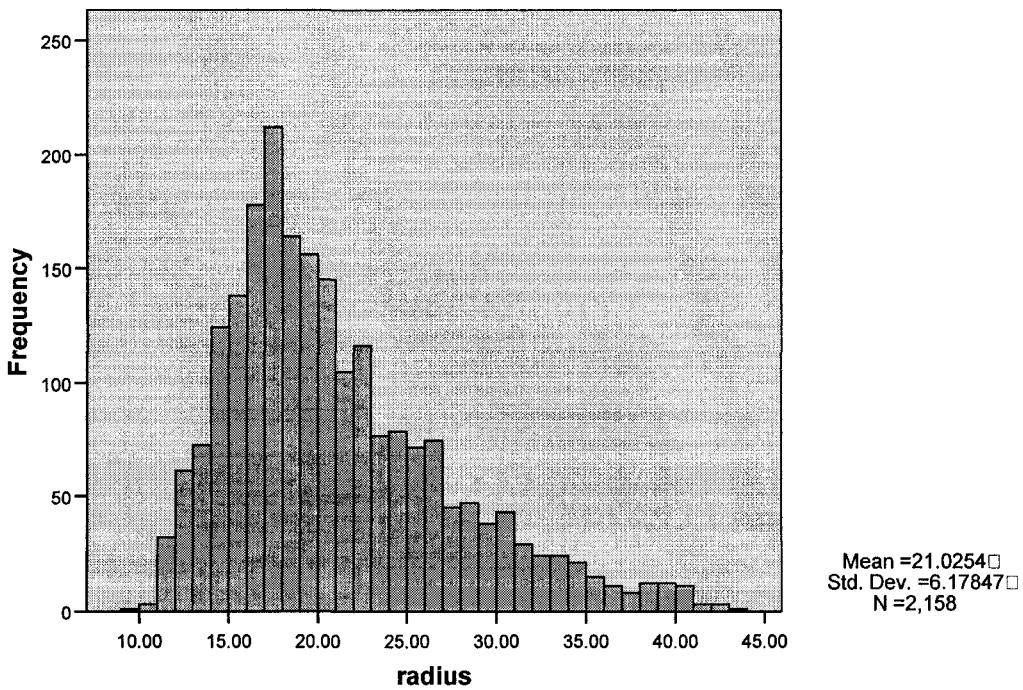
maximum distance between residues is not necessarily the same as the diameter as shown in the figure below. A protein could have three or more equal sides, so choosing one of them as the diameter would create a bounding sphere that did not actually enclose the entire protein.

Given the uneven quality of the data, as many entries in the Protein Data Bank are incomplete or incorrectly formatted, it seemed reasonable to discard values that were more than three standard deviations from the mean for either radius or distance, or that had impossibly small values, leaving N=2158. For distance, the mean for the set was 36.9057, with an SD = 10.06947. For radius, the mean was 21.0254, with an SD = 6.17847. A scatter plot and histogram plotting distance versus number of residues and showing frequency of distance values, respectively, suggest that the data is skewed. The scatter plot and histogram for radius is similar, but not included. This suggests that there is a hard floor for how small a protein with a given number of residues can be. Ideally, the data would show a clear normal distribution and it would be possible to accomplish this through a transformation around the mean. However, given the large N of the data set, it is reasonable to assume that the untransformed results are accurate. (reference)

Looking at which residues are furthest from the sphere is another way of using the available data. In this case, knowing which kind of secondary structure might be on the outside of the molecule could be useful information when doing a simulation, either by limiting the regions chosen as starting points or by suggesting what kind of secondary structure a region that ends up in that position during a simulation could be.



Histogram



The loop information, as shown in the table below, strongly suggests (85% chance) that the residue on the outside of the protein will be in a loop secondary structure, which makes sense since these are the most flexible areas of the protein and more easily make sharp turns back toward the center of the protein molecule. This suggests that looking at loop regions when choosing the point in a simulated structure that will be furthest from the center might be reasonable.

Furthest residue from center in loop (loopa) * Next furthest residue from center in loop (loopb) Crosstabulation

			loopb		
			fals	true	Total
loopa	fals	Count	245	89	334
		Expected Count	99.7	234.3	334.0
	true	Count	399	1425	1824
		Expected Count	544.3	1279.7	1824.0
Total		Count	644	1514	2158
		Expected Count	644.0	1514.0	2158.0

Fig. ?

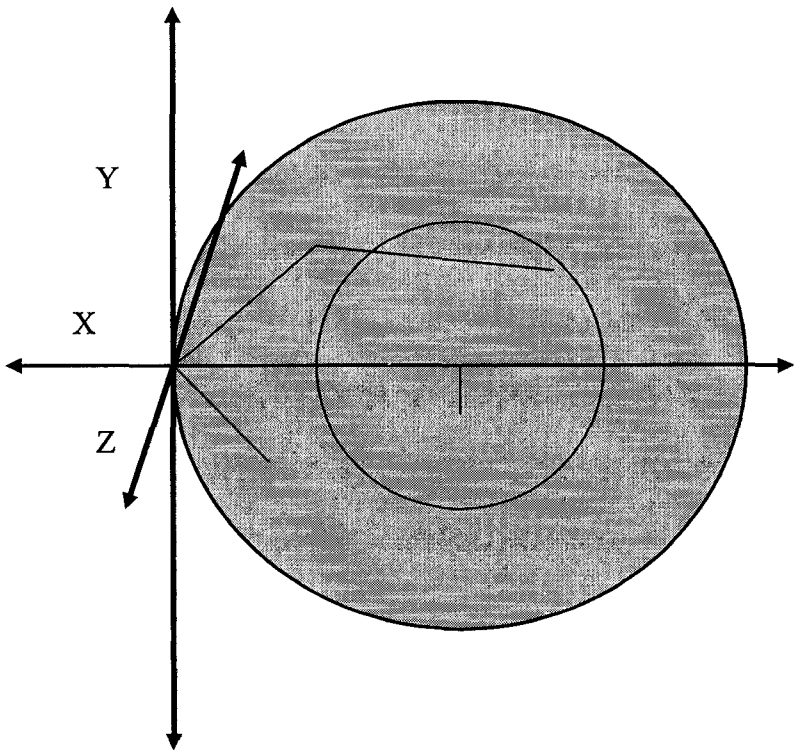
The histograms and scatter plots for distance and radius suggest a linear relationship between the two variables and the number of residues, so linear regression was used to give predictions of the ranges of these variables. Though the data is not normally distributed, violating one of the assumptions of linear regression, the sample size is so large that, according to the central limit theorem, its normalized scores would approach the normal distribution. In this case, since we wanted real values for our prediction intervals, we chose not to transform the data as its unclear what transforming the prediction intervals back to real values means. In any case, the relationship between the two variables and the number of residues is highly significant with a p value of $< .0001$ for both. Using the regression line, a prediction interval that will capture 95% of the

cases can be constructed. Over the range of residues that we are looking at the prediction interval has a range of about 40 angstroms. For example, a protein with 68 residues would be expected to range between 19 and 57 angstroms.

7. Methodology

The experiment focused on setting up a method that would quickly create candidate structures. Ideally, this could be accomplished without spending a lot of time rejecting possible candidates and the resulting configuration space would yield results comparable to those found using a configuration space based on modifying the native or fully extended configurations when submitted to a more refined simulation program.

Configurations were created by choosing discrete values from both the radius and distance ranges, decrementing by a program parameter (set from 2.5 Angstroms, the smallest distance across a residue, to 20) across the entire range of values. A sphere of the chosen radius was given a center point of (0, 0, radius), as seen in the figure below.



For this experiment, the protein was treated as a collection of secondary structure elements rather than individual residues, primarily because this greatly decreases the computation time while still providing a way to test the idea. Each candidate structure was built in one or two chunks, starting with either the first secondary structure element or the first loop secondary structure element. Each chunk was built by adding on secondary structure elements with randomly assigned values for their degrees of freedom until the chunk reached a length exceeding the maximum distance chosen for that iteration, at which point the structure would be encouraged to reverse direction by weighting the vector angle towards an acute value. The degrees of freedom for each link were the vector angle between the current and previous vectors, the dihedral angle consisting of the planes formed by the current and prior two vectors, the twist angle for alpha helix and beta sheet secondary structures and the length for loop secondary structures.

Random values for each degree of freedom were normally distributed around the actual values for the native state of the protein. Structures that either collided with themselves or with the bounding sphere were rejected and the program attempted to create a new structure until hitting the limit on the number of attempts, another program parameter.

Once the chunks were created, they were fitted together using randomly chosen vector and dihedral angles, again based on the native state, which tried to minimize the distance between the two vectors while avoiding self-collision or collision with the bounding sphere by using a weight factor to make the vector angle tend to be more acute after each attempt. Once a configuration was created, its energy level was checked and the

configuration was accepted if it was below a constant threshold. This was repeated for each combination of radius and distance steps possible in the prediction interval.

Statistics on the number of configurations accepted or rejected for each set of parameters was collected and saved in a file. The configurations themselves were saved in a format that allowed them to be used as seed values for a Monte Carlo simulation program, used in Apaydin, et al's work. The program itself borrowed heavily from source code used in Apaydin's work and the work of Itay, Schwarzer and Latombe who used bounding volumes to simplify the computation of total energy in a Monte Carlo simulation. (Lotan 2003) The simulations were run on a Pentium Celeron D 3.46Ghz system with 2Gb of memory.

Listed below is the pseudo-code for the main loop and function of the program that implements and tests the ideas in this paper.

Figure N. Pseudo-code for main algorithm and buildChain function

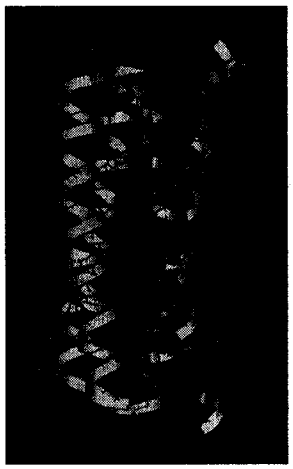
```
For radius = max to (native radius – step); step- -
 For distance = max to (max distance – step); step- -
  for each secondary structure element = type LOOP
   while ! Accepted
    if LOOP element index > 0
      chain1 = buildChain(0, LOOP index);
      chain2 = buildChain(LOOP index + 1, total elements);
      combineChains(chain1, chain2);
    else
     chain = buildChain(0, total elements);
        } // if
If (! sphereCollide(chain))
      if (!selfCollide(chain))
        accepted = true;
      else
```

```
        accepted= false;
      else
        accepted = false;
    }
        } // while
  } // for


BuildChain(int firstIdx, int lastIdx)
For each sse element firstIdx to lastIdx
  While (! Accepted)
   if firstIdx
    set base = origin;
   else
    set base = lastElement->end;
   paramers = changeParameters(vector, dihedral,
                        length, twist);
   newSse = computeVector(parameters);
   if (! Sphere Collide(newSSe) &&
         ! selfCollide(newSse))
    accepted = true;
   else
    accepted = false;
  } while
 } for
```

For this experiment, two proteins were used as the basis of the simulations, RNA modulator (1ROP) and Engrailed Homeodomain (1HDD). 1ROP has 56 residues and its native state consists of three secondary structures, two alpha helices and a small loop. 1HDD has 57 residues and five secondary structures, a mixture of all three types. The information on their native state's secondary and tertiary structures was retrieved from the protein data bank and filtered through several routines, including the DSSP program by Kabsch and Sander (Kabsch and Sander, 1983), which puts secondary structure function into an easily processed format.

Simulated Proteins



RNA modulator (1ROP)

Engrailed Homeodomain (1HDD)

Figure n.

For each protein, the roadmap construction program was run once with each combination of the following run-time parameters:

- Amount to decrement radius and distance at each step (3, 6, 9, 12 angstroms)
- Whether or not to focus on loop structures as a starting point (boolean)
- Number of attempts to make for each combination of radius, distance and starting point (25 – 75 by 10's)

8. Results

9. Future Direction

The field of protein structure simulation is large and expanding daily, but the robotic approaches, especially the probabilistic path planning approaches, offer the possibility of doing simulations more quickly by using a probabilistically complete rather than an empirically complete approach. The main variations of the approach use it either to explore paths in detail or to explore a protein's structure probabilistically using Markov Chain methods.

The technique described in this paper adds a constraint that can be used by other techniques to limit their configuration space. This is well-suited to a path-planning approach, which already works from a randomly distributed configuration space, but could be used as a method of seeding start configurations for other approaches. An obvious extension is to create an application that uses residues, rather than secondary structure elements, to build configurations. This could be plugged into a much wider variety of techniques.

Another way to use this approach would be as the basis for an ab initio configuration space for random path planning. Rather than working from the extended configuration to the native state, a path could be calculated from acceptable configurations at the maximum radius to acceptable configurations at lower radii.

Finally, the technique suggests a different way of using the data already available on protein structure. By looking for significant patterns and relationships between characteristics of proteins and data, different possibilities for how to constrain the configuration space might be suggested that will further decrease the effort required to do a simulation.

Figures

Figure 1. *Illustration adapted from the National Human Genome Research Institute (NHGRI) Genetic Illustrations entry for mRNA, http://www.ncbi.nlm.nih.gov/Class/MLACourse/Modules/MolBioReview/images/central_dogma.gif*

Figure 2. Protein chain diagram, http://wiz2.pharm.wayne.edu/biochem/PRO4.GIF.

Figure 3. Ramachandran plot example, http://femto.cs.uiuc.edu/~sbond/reports/villin/rama.gif

Figure 4. Boltzmann Distribution diagram, http://www.webchem.net/notes/how_far/kinetics/maxwel2.gif, 2005

Figure 5. Free energy landscape http://cnx.org/content/m11467/latest/funnel.jpg, Lydia Kavraki

Figure 6. Lattice Model Diagram, dimacs.rutgers.edu/ ~newmana/prot.gif.

Figure 7. Bounding volume diagram http://wwwvis.informatik.unistuttgart.de/img/sommer/Autobench/BoundingVolumeHierachyLarge.gif

Figure 8. Sample Voronoi Diagram, http://www.cs.utexas.edu/users/amenta/powercrust/unionspix/voronoi.gif

Figure 9. Robot arm diagram, http://www.ranchbots.com/robot_arm/images/arm_diagram.jpg

References

Amato, N. M., K. A. Dill, et al. (2003). "Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures." Journal of computational biology: a journal of computational molecular cell biology **10**(3-4): 239-55.

Amato, N. M. and G. Song (2002). "Using motion planning to study protein folding pathways." Journal of computational biology: a journal of computational molecular cell biology **9**(2): 149-68.

Apaydin, M. S., D. L. Brutlag, et al. (2003). "Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion." Journal Of Computational Biology **10**(3-4): 257-281.

Apaydin, M. S., C. E. Guestrin, et al. (2002). "Stochastic roadmap simulation for the study of ligand-protein interactions." Bioinformatics **18**: S18-S26.

Bergeron, B. (2003). "Bioinformatics Computing."

Branden, C. a. T., J. (1991). "Introduction to Protein Structure."

Canutescu, A. and Dunbrack, Jr., R. L. (2003). "Cyclic coordinate descent: A robotics algorithm for protein loop closure." Protein Science 12:963-972.

Challou, D., Boley, D., Gini, M., Kumar, V. and Olson, C. (1998). "Parallel Search Algorithms for Robot Motion Planning, in Gupta and del Pobil, Practical Motion Planning in Robotics." 115 -132.

Clote, P. a. B., R. (2000). "Computational Molecular Biology: An Introduction."

Demaine, E. D. L., S.; O'Rourke, J. (2003). "Geometric Restrictions on Producible Polygonal Protein Chains." Algorithmica **to be published**.

Gupta, K. a. d. P., A.P. (1998). "Practical Motion Planning in Robotics: Current Approaches and Future Directions."

Kabsch,W. & Sander,C. 1983 "Dictionary of protein secondary structure: Pattern recognition of hydrogen bonded and geometrical features", Biopolymers 22:2577-2637

Kavraki, L. E. a. Latombe., J.C. (1998). "Probabilistic Roadmaps for Robot Path Planning, in Gupta and del Pobil, Practical Motion Planning in Robotics." 33-54.

Kirkpatrick, S. G. J., C.D>; and Vecchi, M. P. (1983). "Optimization by simulated annealing." Science(220): 671-680.

Lau, K. F. D., K.A. (1989). "A lattice statistical mechanics model of the conformational and sequence spaces of proteins." Macromolecules(22): 3986-3997.

Lotan, I. S., F.; Latombe, J. C. (2003). "Efficient Energy Computation for Monte Carlo Simulation of Proteins." 3rd Workshop on Algorithms in Bioinformatics (WABI).

Mirtich, B. "Efficient Algorithms for Two-Phase Collision Detection, in Gupta and del Pobil, Practical Motion Planning in Robotics." 204-224.

Murphy, R. R. (2000). "Introduction to AI Robotics."

Ramachandran, G. N. S., V. (1968). "Conformation of polypeptides and proteins." Adv. Prot. Chem.(28): 283-437.

Rost, B. (2001). "Review: Protein Secondary Structure Prediction Continues to Rise." Journal of Structural Biology(134): 204-218.

Singh, P. C. L., J.C.; and Brutlag, D.J. (1999). "A Motion Planning Approach to Flexible Ligand Binding." Proc. 7th Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB): 252-261.

Sippl, M. (1990). "Calculation of conformation ensembles from potentials of mean force." Journal of Molecular Biology(213): 859-883.

van den Bedem, H. L., I.; Latombe, J.; and Deacon, A.M. (2005). "Real-space protein-model completion: an inverse-kinematics approach." Biological Crystallography **61** (Part 1): 2-13.