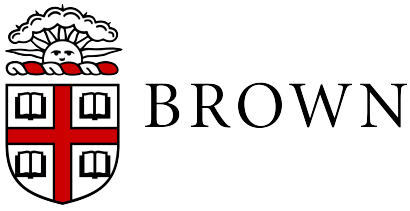
Protein Folding Prediction and Visualization Techniques Based on Hydrophobic Side Chain Interactions

Adrian Stefan Turcu



Computer Science Department
Advisor: Professor Sorin Istrail
Second Reader: Dr. Nicola Neretti
Brown University

May 2019

Contents

Acknowledgments

I am very grateful for all of the professors that helped me with this thesis and throughout the entire research process. Thank you to Sorin, who willingly accepted me as a research assistant in my sophomore year and who has been a great friend and teacher ever since. Thank you to Tim Nelson for his SAT solver insights, as well as his support as I tried to better understand this topic. Lastly, thank you to Dr. Nicola Neretti for agreeing to be the second reader for my thesis.

I want to thank my family, who has always been there for me and encouraged me to do my best. I honestly do not know where I would be without your help and guidance. I also want to thank my friends, who have made the past few years the best of my life. I will never forget all the times we laughed together and supported each other, and I am beyond lucky to have met you.

1 Abstract

Proteins, and the amino acids that form them, are some of the fundamental building blocks of life. The amino acids come together and fold in very precise ways to form units capable of performing specific functions. Because of the immense importance of proteins, understanding how they fold is essential. This is especially true in the case of protein misfolding, which can lead to a variety of diseases such as Alzheimer's Disease and prion diseases. By developing a way to predict how proteins fold, the harmful impacts of these diseases could be mitigated. For this Honors Thesis, an attempt is made to develop a deeper understanding of the driving force behind protein folding. More specifically, the role that hydrophobic interactions between amino acids play in determining folding patterns is examined via Protein Database (PDB) file visualization, backtracking algorithms, and SAT solver techniques. Through examination of the PDB images generated, clustering trends for hydrophobic side chains were observed. This insight was followed by the development of algorithms for simplified 2D square lattice and 3D cubic lattice models of proteins that were aimed towards maximizing contact between hydrophobic amino acids. These algorithms were successful in their goal for these basic representations, but many limitations were encountered. To better model and predict protein folding, more complex lattices should be explored, along with a larger number of amino acids.

2 Background

2.1 Protein Structure:

All proteins are made up of sequences of molecules called amino acids. There are twenty amino acids, which can vary greatly in size and characteristics. The general structure of an amino acid is shown below:

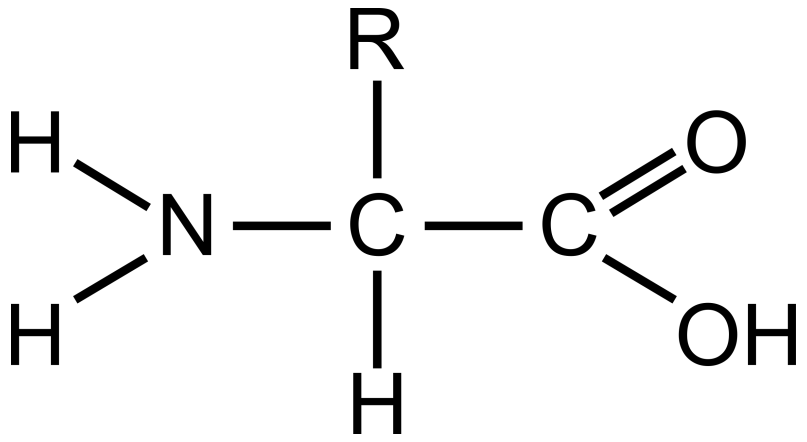


Figure 2.1.1: General Amino Acid Structure

The N, C, H, and O letters correspond to nitrogen, carbon, hydrogen, and oxygen. The portion of the amino acid with the nitrogen and two hydrogens is called an amine group

and the part with a carbon bonded to the two oxygens is called a carboxyl group. The R section is a variable group that changes based on amino acid. The R group is key in giving amino acids different properties. For instance, a nonpolar R group makes the amino acid hydrophobic, as it is not soluble in water and tends to avoid it. Because of these unique characteristics, different sequences of amino acids fold in very different ways. Varying folding patterns in amino acid sequences result in complex proteins with highly specific functions. Understanding what exactly drives this folding, and how it can be predicted from just the amino acid sequence, are very important problems in the field of biology.

2.2 Previous Explorations into Protein Folding

There have been many attempts made in developing a better understanding of how proteins fold. Because working with this issue in open 3D space poses significant computational barriers, the protein folding problem has been simplified to 2D and 3D lattice models. One approach that has been rigorously studied, and which is the primary model explored in this paper, is the Hydrophobic-polar (HP) model proposed by Ken Dill. In this model, amino acids are designated as either hydrophobic or polar, and they must be placed on a lattice in a "self-avoiding walk." This means that the amino acids must be placed on the lattice without overlap. In addition to this, this model is built around the idea that protein folding is driven by hydrophobic interactions, and thus it seeks to maximize the number of hydrophobic side chains that are adjacent to one another [4, 5, 6, 7]. Although this protein folding problem has been proven to be NP-hard, multiple algorithms exist that seek to optimize amino acid placement on simple lattices [3, 8]. With this model, some exploration has been done in the realm of 2D and 3D hexagonal lattices with diagonals as well [9, 10]. One major goal is to develop an accurate and efficient way to predict simplified protein folding on the face-centered cubic, or FCC lattice. The FCC lattice, shown below, most closely mimics the actual structure of proteins. Thus, having an effective approach for solving the protein folding problem on this model would be of tremendous importance to the field of computational biology [11].

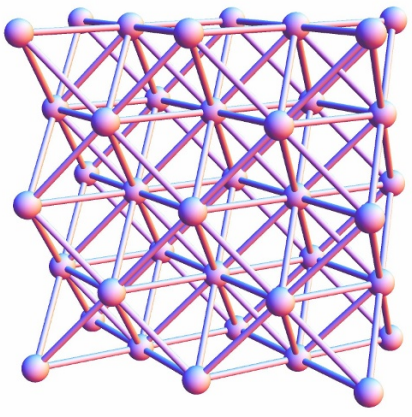


Figure 2.2.1: FCC Lattice

2.3 Protein Misfolding Diseases

Developing techniques to predict how proteins will fold holds much promise for the medical field, as there are several diseases that can occur due to misfolding of proteins. For instance, Alzheimer's disease is triggered by abnormalities in the tau and β-amyloid proteins, causing them to misfold. This starts a series of events that leads to a buildup of neurofibrillary tangles and β-amyloid plaques that harm the brain, thus causing the symptoms of Alzheimer's [1]. Prion diseases are harmful in a similar fashion, as they arise when misfolded proteins called prions are created in the body. More specifically, proteins that have numerous α-helical structures will refold to instead possess β-sheets, which can drastically impact protein function. Some examples of prion diseases are Creutzfeldt-Jakob disease and fatal familial insomnia [2]. These diseases, along with Alzheimer's, pose a very serious threat, and there is still no completely effective cure for them. If accurate protein folding prediction algorithms are developed, however, a much better understanding of how and why protein misfolding occurs could be gained. This would be a great step forward not only in the realm of computer science, but also in medicine and biology.

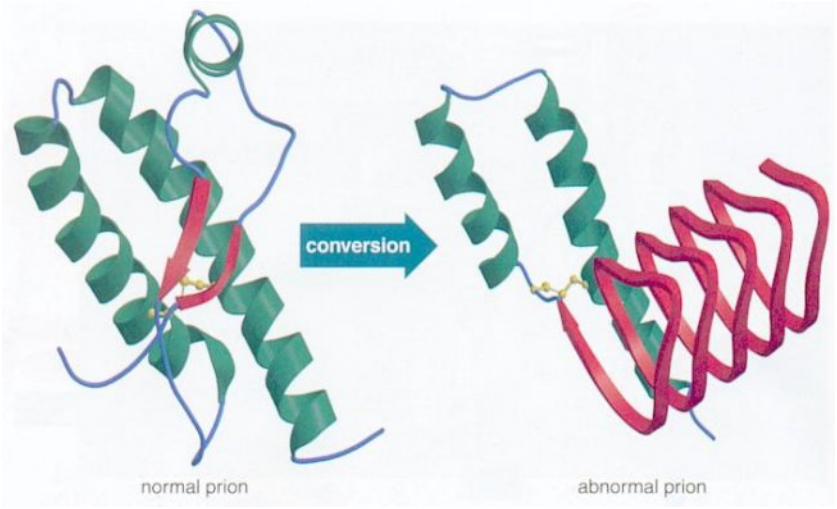


Figure 2.3.1: Prion Misfolding

3 Protein Database Visualization

3.1 Protein Database

The files used for creating protein visualizations were downloaded from the Protein Database, or PDB. Every protein has a file in this database that highlights its key characteristics, and lists every atom making up the protein. Each row describing an atom contains the 3D coordinates of the atom, its element, the type of amino acid it is a part of, and other identifying information. This portion of the PDB file is important when constructing visualizations for a protein, and thus a parsing tool was created to extract the desired information.

```
ATOM      7  N   CYS A   3      38.240   2.099  -5.150  1.00  5.05           N
ATOM      8  CA  CYS A   3      38.673   3.471  -5.402  1.00  4.41           C
ATOM      9  C   CYS A   3      39.052   3.647  -6.869  1.00  4.23           C
ATOM     10  O   CYS A   3      39.919   2.945  -7.390  1.00  4.59           O
ATOM     11  CB  CYS A   3      39.880   3.821  -4.530  1.00  4.15           C
ATOM     12  SG  CYS A   3      39.450   3.592  -2.787  1.00  4.20           S
ATOM     13  H   CYS A   3      38.846   1.478  -4.695  1.00  5.43           H
ATOM     14  HA  CYS A   3      37.863   4.144  -5.165  1.00  4.68           H
ATOM     15  HB2 CYS A   3      40.707   3.175  -4.784  1.00  4.37           H
ATOM     16  HB3 CYS A   3      40.160   4.850  -4.700  1.00  4.24           H
ATOM     17  N   GLY A   4      38.392   4.595  -7.528  1.00  4.02           N
ATOM     18  CA  GLY A   4      38.666   4.861  -8.937  1.00  4.00           C
ATOM     19  C   GLY A   4      39.468   6.147  -9.099  1.00  3.54           C
ATOM     20  O   GLY A   4      40.471   6.185  -9.812  1.00  3.77           O
ATOM     21  H   GLY A   4      37.712   5.124  -7.060  1.00  4.14           H
ATOM     22  HA2 GLY A   4      39.226   4.036  -9.352  1.00  4.34           H
ATOM     23  HA3 GLY A   4      37.731   4.960  -9.469  1.00  4.26           H
ATOM     24  N   GLY A   5      39.016   7.201  -8.426  1.00  3.35           N
ATOM     25  CA  GLY A   5      39.698   8.490  -8.498  1.00  3.10           C
ATOM     26  C   GLY A   5      39.772   9.140  -7.121  1.00  2.76           C
ATOM     27  O   GLY A   5      39.711  10.362  -6.992  1.00  2.70           O
ATOM     28  H   GLY A   5      38.212   7.111  -7.872  1.00  3.68           H
ATOM     29  HA2 GLY A   5      40.700   8.340  -8.877  1.00  3.31           H
ATOM     30  HA3 GLY A   5      39.158   9.141  -9.168  1.00  3.18           H
ATOM     31  N   VAL A   6      39.903   8.306  -6.093  1.00  2.58           N
ATOM     32  CA  VAL A   6      39.984   8.808  -4.724  1.00  2.27           C
ATOM     33  C   VAL A   6      41.330   9.486  -4.486  1.00  2.12           C
ATOM     34  O   VAL A   6      41.404  10.559  -3.887  1.00  1.92           O
ATOM     35  CB  VAL A   6      39.816   7.664  -3.720  1.00  2.23           C
ATOM     36  CG1 VAL A   6      39.702   8.236  -2.305  1.00  2.28           C
ATOM     37  CG2 VAL A   6      38.547   6.874  -4.051  1.00  2.62           C
ATOM     38  H   VAL A   6      39.946   7.341  -6.257  1.00  2.69           H
ATOM     39  HA  VAL A   6      39.196   9.529  -4.567  1.00  2.24           H
ATOM     40  HB  VAL A   6      40.674   7.009  -3.774  1.00  2.55           H
ATOM     41 HG11 VAL A   6      39.046   9.094  -2.316  1.00  2.59           H
ATOM     42 HG12 VAL A   6      39.298   7.483  -1.644  1.00  2.65           H
ATOM     43 HG13 VAL A   6      40.679   8.535  -1.958  1.00  2.51           H
ATOM     44 HG21 VAL A   6      37.699   7.544  -4.063  1.00  2.87           H
ATOM     45 HG22 VAL A   6      38.654   6.411  -5.021  1.00  3.09           H
ATOM     46 HG23 VAL A   6      38.392   6.111  -3.302  1.00  2.92           H
```

Figure 3.1.1: PDB File Sample

3.2 Grouping by Size and Hydrophobicity

There are many PDB processing software, such as RasMol, that can create accurate 3D images of the protein with every atom clearly shown. To analyze the effect that hydrophobicity and relative size of the amino acids have on protein structure, this approach was modified. The alpha carbons making up the backbone were still shown, but all other backbone elements were stripped away to reduce clutter. Additionally, the side chain groups were represented in a much simpler manner. More specifically, instead of showing every atom in the side chains, one sphere was used to represent the entire side chain. This sphere was given a different color depending on whether it was hydrophobic or not, and it was classified as small, medium, or large based on how many atoms made it up. To place this sphere, the center of mass of all atoms in the side chain was calculated. By taking this approach, the direct effects of hydrophobicity and size of side chains on the protein's overall structure can be seen.

The following scheme was used to classify side chain segments:

- Backbone Carbon = Gray Sphere
- Small Hydrophobic Side Chain (less than 4 atoms) = Orange Sphere
- Medium Hydrophobic Side Chain (at least 4 atoms, less than 6 atoms) = Red Sphere
- Large Hydrophobic Side Chain (at least 6 atoms) = Brown Sphere
- Small Hydrophilic Side Chain (less than 4 atoms) = Pink Sphere
- Medium Hydrophilic Side Chain (at least 4 atoms, less than 6 atoms) = Purple Sphere
- Large Hydrophilic Side Chain (at least 6 atoms) = Magenta Sphere
- Connection Between Backbone Segments = Gray Edge
- Connection Between Backbone and Hydrophobic Segments = Green Edge
- Connection Between Backbone and Hydrophilic Segments = Blue Edge

3.3 File Parsing

From the PDB file, various information was extracted and translated into text files for further processing. To fully visualize a PDB file in the desired fashion, the coordinates of alpha carbons and of the spheres representing side chain groups are essential. However, the side chain spheres must also be divided based on both size and hydrophobicity, which can be determined from looking at what amino acid an atom corresponds to. Lastly, information about how alpha carbons are connected to side chain groups and to each other is vital to make sense of the protein visualization. Once this information was separated into text files, a Mathematica program was written to develop colored 3D images.

3.4 Mathematica Visualizations

Using Mathematica's Graphics3D package, the information pulled from the PDB files was translated into structurally accurate images of proteins with an accent on amino acid size and hydrophobicity. Using the alpha carbon coordinates, small gray spheres were drawn and connected. Then, the coordinates of the side chain spheres were read, and depending on what number of atoms and hydrophobicity they were classified as, they were given a size and color. This visualization was done in many ways and for several proteins. Three examples, shown below, are a myoglobin mutant, an inhibited thrombin, and an aminopeptidase in complex with angiotensin IV.

3.4.1 3rgk (Human Myoglobin Mutant K45R)

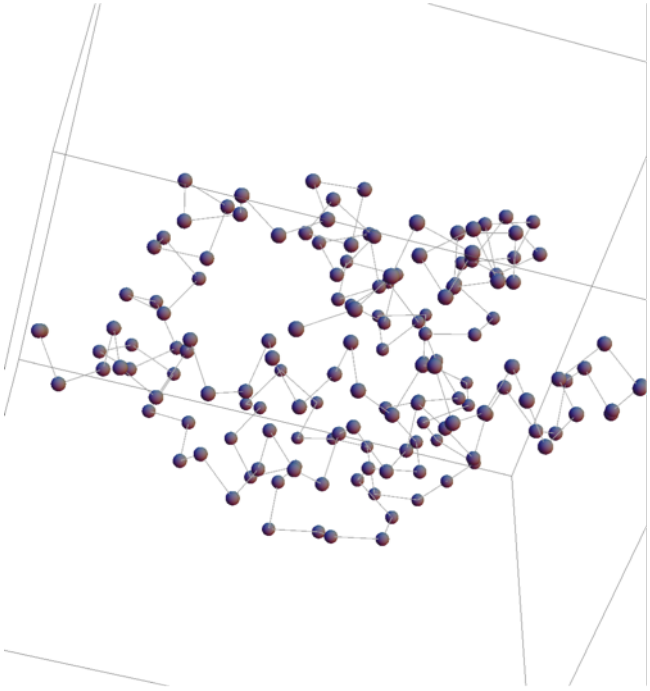


Figure 3.4.1.1: 3rgk: Backbone Only

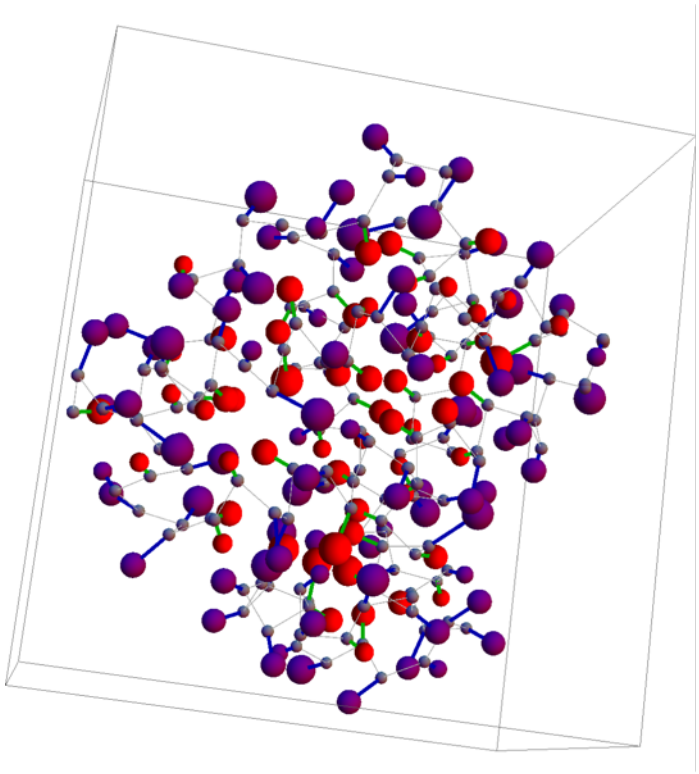


Figure 3.4.1.2: 3rgk: Two-color, Only Red and Purple

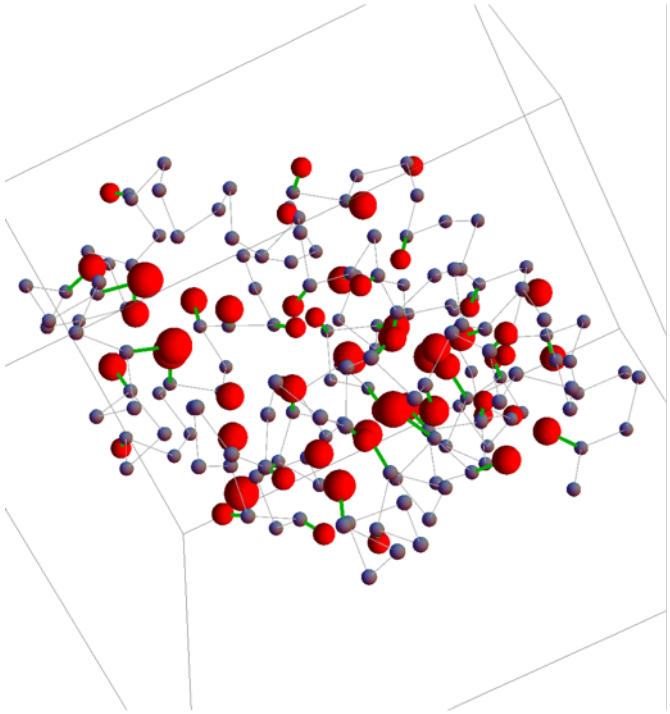


Figure 3.4.1.3: 3rgk: Two-color, Hydrophobic Groups Only

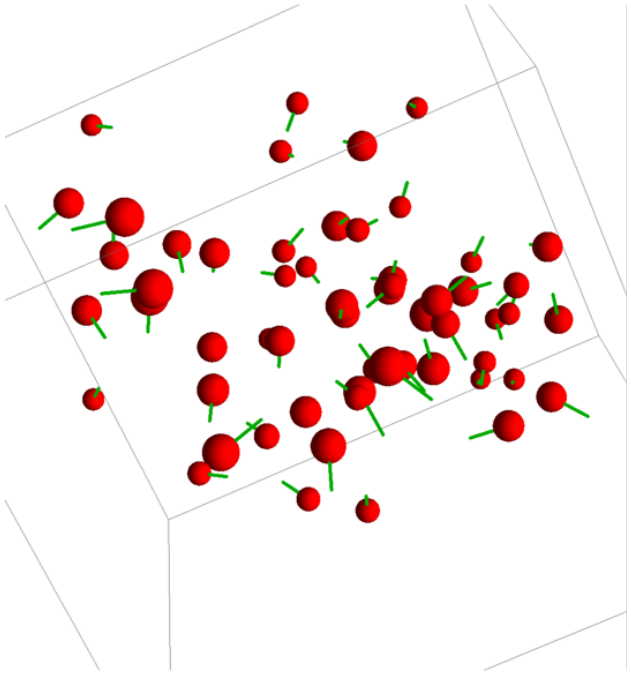


Figure 3.4.1.4: 3rgk: Two-color, Hydrophobic Groups Only, No Backbone

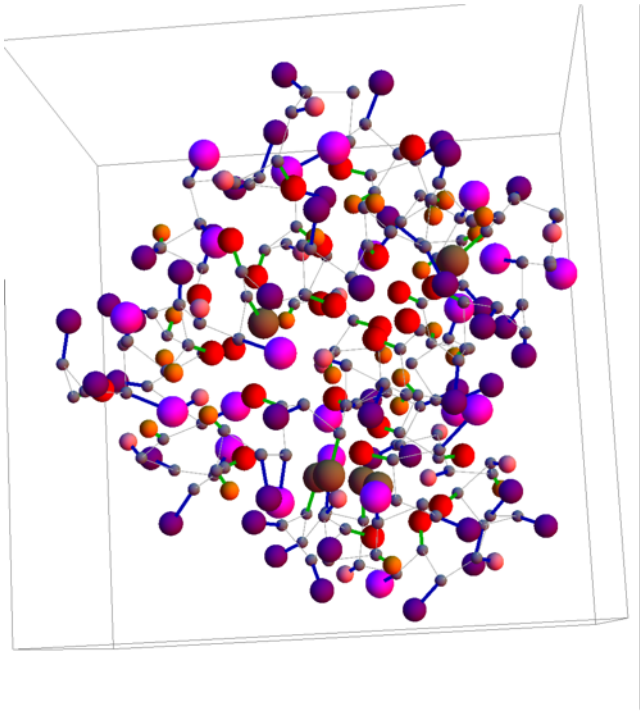


Figure 3.4.1.5: 3rgk: Multi-color

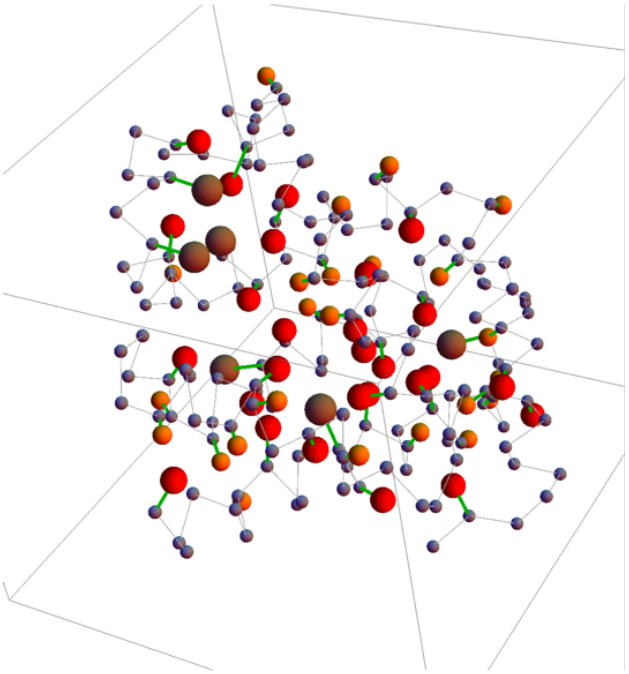


Figure 3.4.1.6: 3rgk: Multi-color, Hydrophobic Groups Only

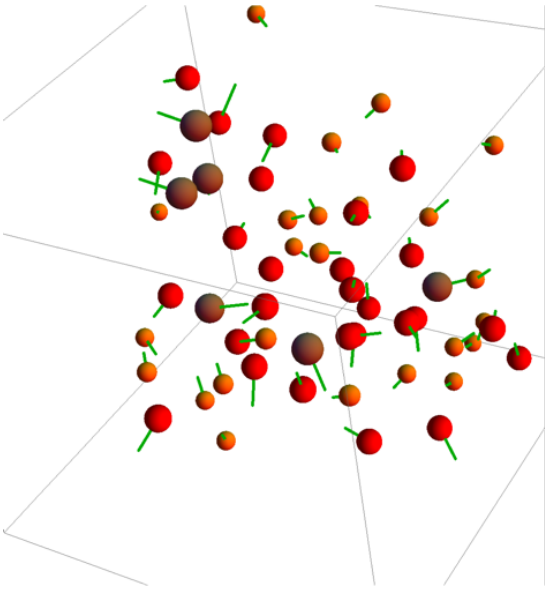


Figure 3.4.1.7: 3rgk: Multi-color, Hydrophobic Groups Only, No Backbone

3.4.2 1a2c (Thrombin Inhibited by AERUGINOSIN298-A from a Blue-Green Alga)

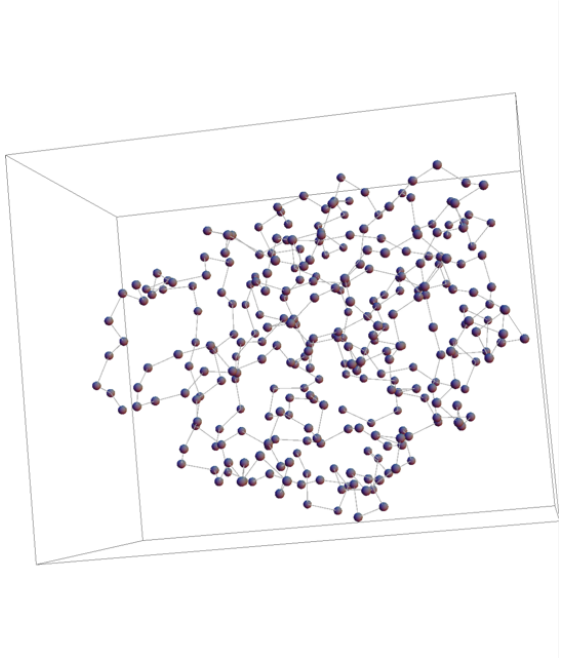


Figure 3.4.2.1: 1a2c: Backbone Only

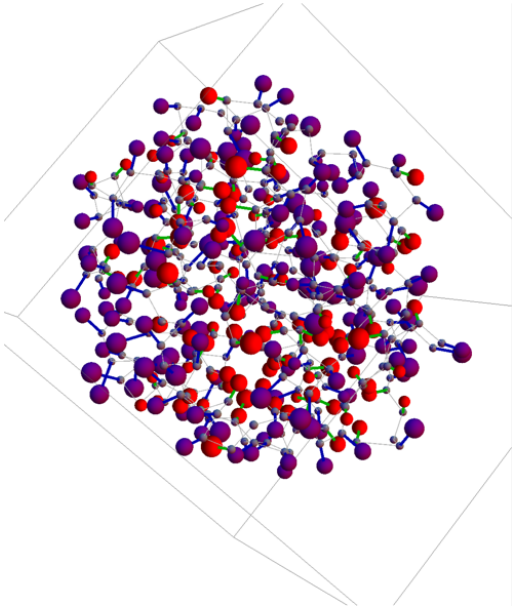


Figure 3.4.2.2: 1a2c: Two-color, Only Red and Purple

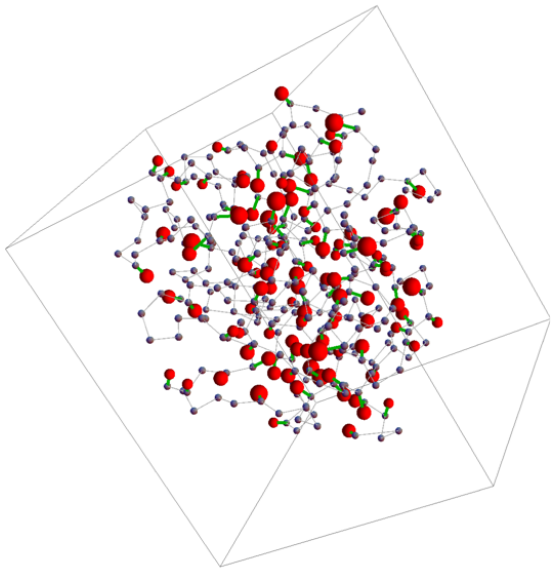


Figure 3.4.2.3: 1a2c: Two-color, Hydrophobic Groups Only

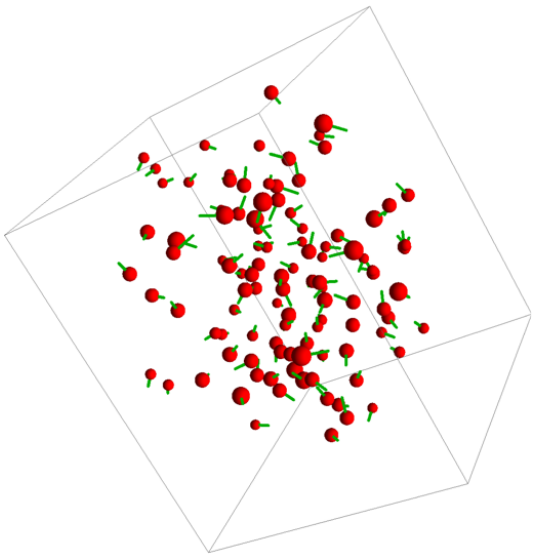


Figure 3.4.2.4: 1a2c: Two-color, Hydrophobic Groups Only, No Backbone

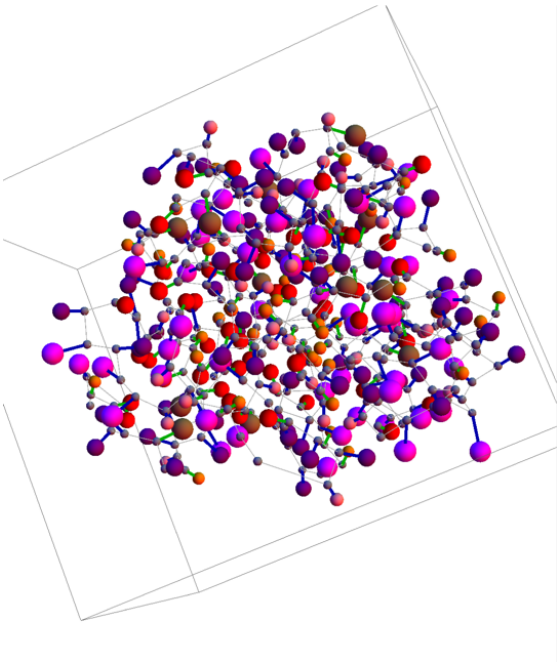


Figure 3.4.2.5: 1a2c: Multi-color

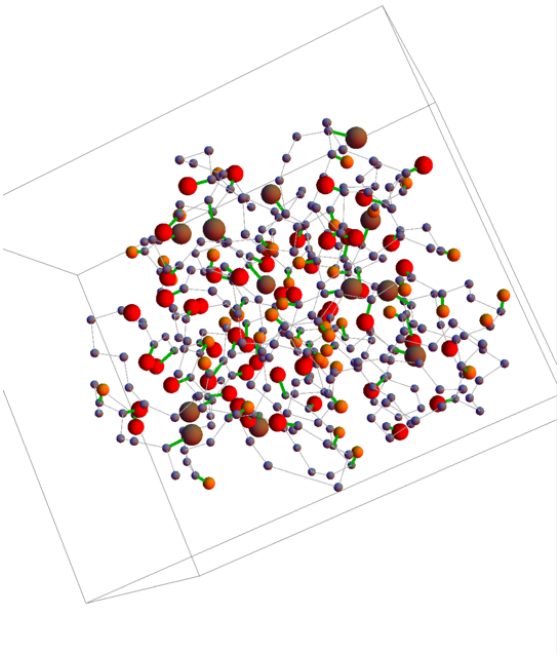


Figure 3.4.2.6: 1a2c: Multi-color, Hydrophobic Groups Only

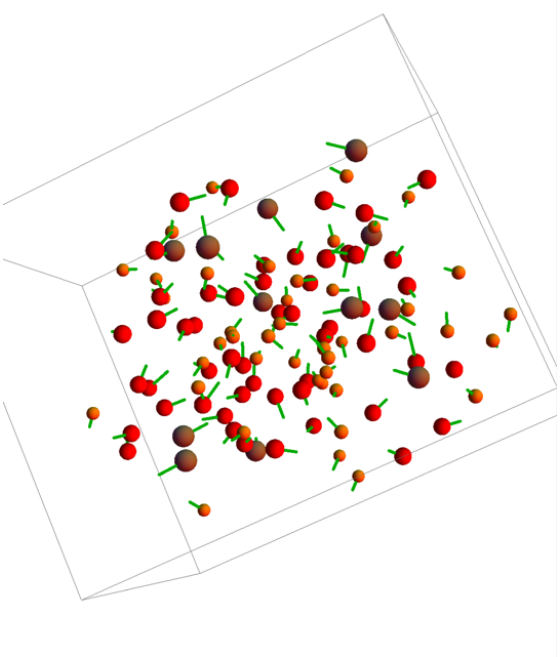


Figure 3.4.2.7: 1a2c: Multi-color, Hydrophobic Groups Only, No Backbone

3.4.3 4fys (Human Aminopeptidase N (CD13) in Complex with Angiotensin IV)

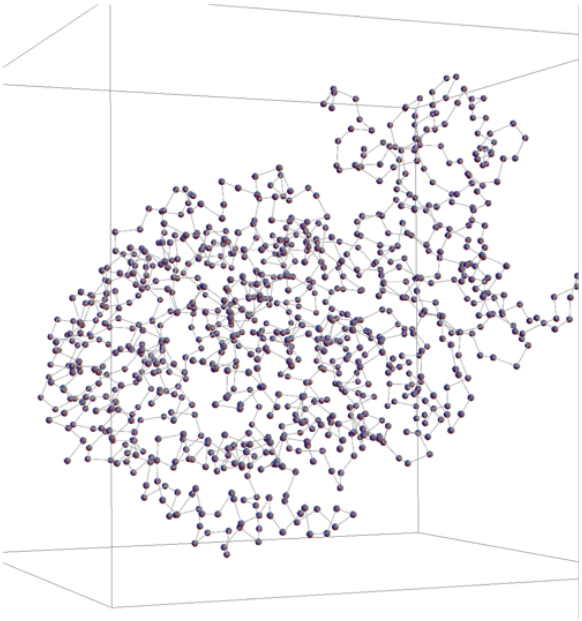


Figure 3.4.3.1: 4fys: Backbone Only

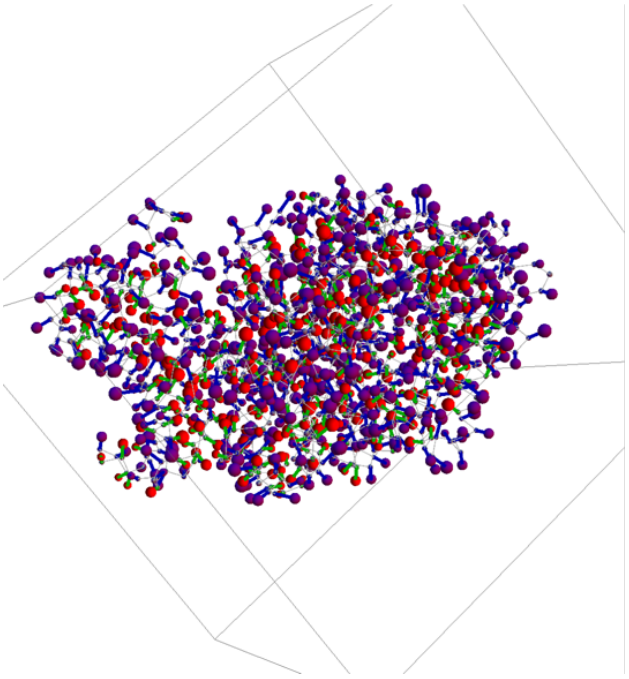


Figure 3.4.3.2: 4fys: Two-color, Only Red and Purple

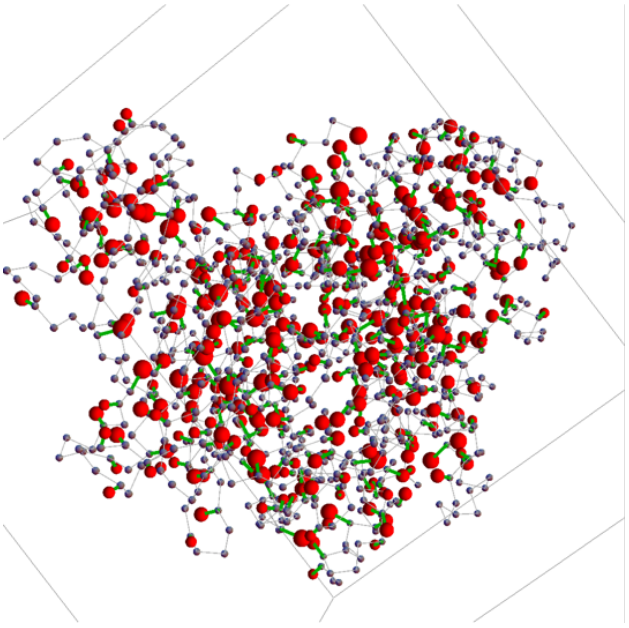


Figure 3.4.3.3: 4fys: Two-color, Hydrophobic Groups Only

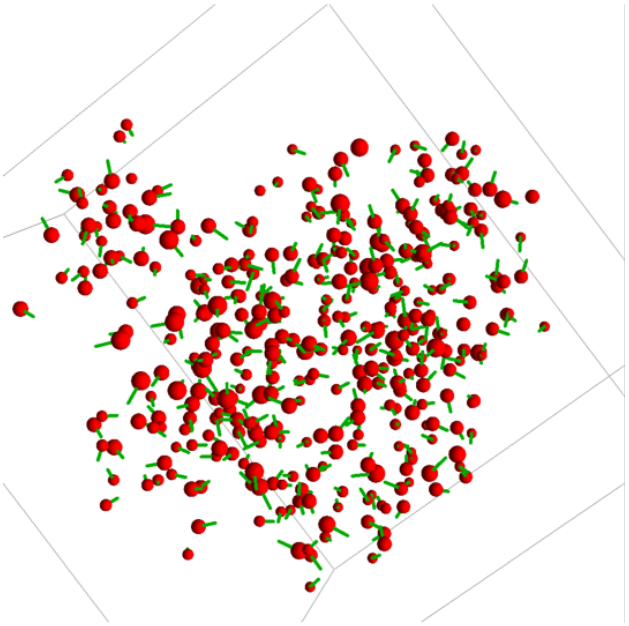


Figure 3.4.3.4: 4fys: Two-color, Hydrophobic Groups Only, No Backbone

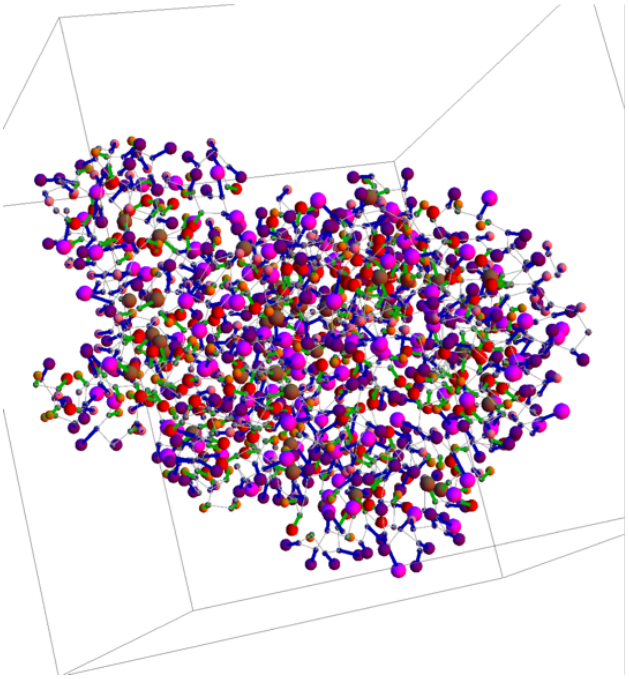


Figure 3.4.3.5: 4fys: Multi-color

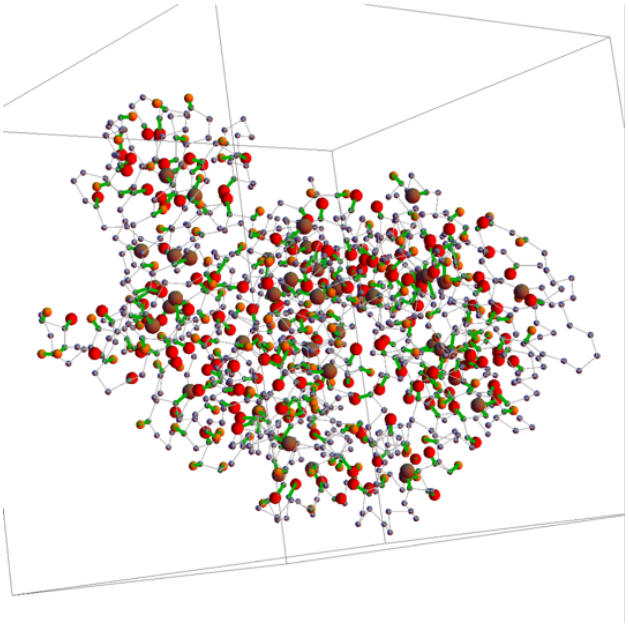


Figure 3.4.3.6: 4fys: Multi-color, Hydrophobic Groups Only

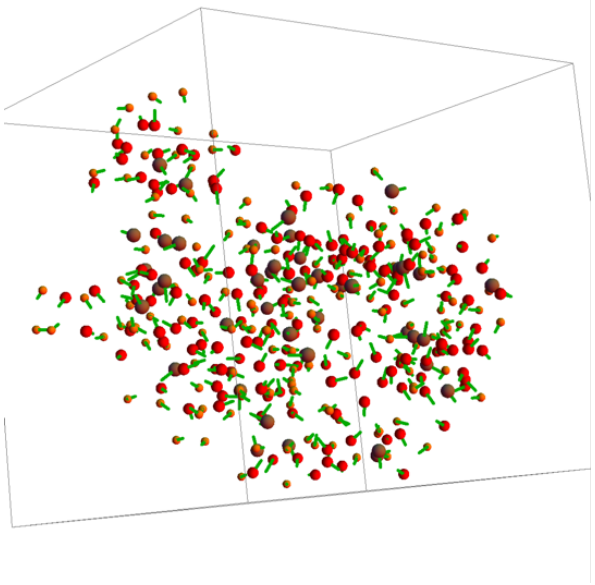


Figure 3.4.3.7: 4fys: Multi-color, Hydrophobic Groups Only, No Backbone

3.5 Insights

From the images generated, a few insights were developed concerning the nature of hydrophobic interactions in proteins. Specifically, it seems that hydrophobic side chains tend to point inwards and cluster together in the center of the protein, away from the aqueous surroundings. These findings support the currently accepted idea that protein folding is built around hydrophobic interactions at the core of the protein. Some trends related to amino acid size were also observed, but they should be confirmed with more analysis. Namely, larger hydrophobic amino acids tend to group together in some scenarios, and also seemed to have a stronger tendency to remain in the center of the protein than smaller hydrophobic groups. Using hydrophobic interactions between amino acids as the assumed driving force of protein folding, algorithms were developed to predict folding patterns for proteins in simplified models.

3.6 Future Work

The program created to extract data from the PDB files can be augmented in many ways to get more information about amino acid interactions. One approach that was recently explored is adding the ability to calculate the exact distance between amino acid groups. By doing this, the hope is that trends will be observed that reveal how side chain hydrophobicity affects amino acid interactions and thus protein folding and structure. Additionally, from a list of distances between amino acids, energy levels corresponding to these interactions can be computed. Knowledge of these values could open the door for minimization or maximization problems that may ultimately be used for protein folding prediction. Lastly, a greater number of proteins should be visualized using this software so that more evidence can be gathered for the trends observed.

4 Backtracking Algorithm

4.1 Representation and Problem

Before discussing the problem, the concepts of a "bisphere segment" and a "contact" must be defined. Proteins are very complex molecules that are made up of a repeated backbone segment, which, through the backbone alpha carbon, is bound to various R groups, or amino acid side chains. Creating algorithms for something as complicated as this is difficult, so a simplified representation was used. Namely, each protein was considered as a sequence of bisphere segments. Each bisphere segment consists of two spheres, one corresponding to a backbone unit and another corresponding to the connected hydrophobic amino acid side chain, as well as an edge between them. The side chain spheres in these segments form a "contact" when they are adjacent to each other in the grid structure being used. The first protein folding problem that was tackled is as follows: given a 2D rectangular lattice with dimensions x by y, and given some original placement of n bisphere segments on the grid, what is maximum number of contacts that can be made by only switching the positions of the backbone and side chain spheres in the segments? In other words, given the only modification that is allowed is inverting each individual bisphere segment, how can the number of adjacent side chain spheres be maximized? This problem was also expanded to 3D, where a 3D cubic lattice of dimensions x by y by z was used. Example inputs and outputs for these problems are shown below, where the green circles and spheres represent backbone segments, and the red circles and spheres represent hydrophobic side chain segments.

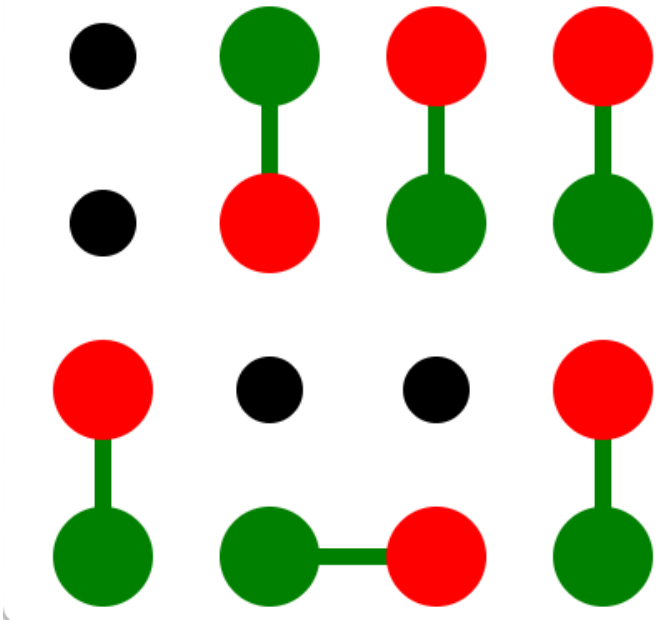


Figure 4.1.1: 2D Input (1 Contact)

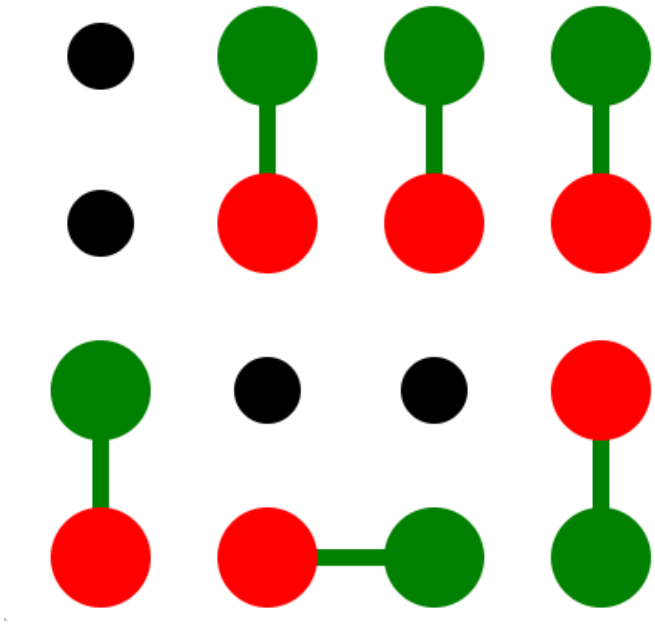


Figure 4.1.2: 2D Output (4 Contacts)

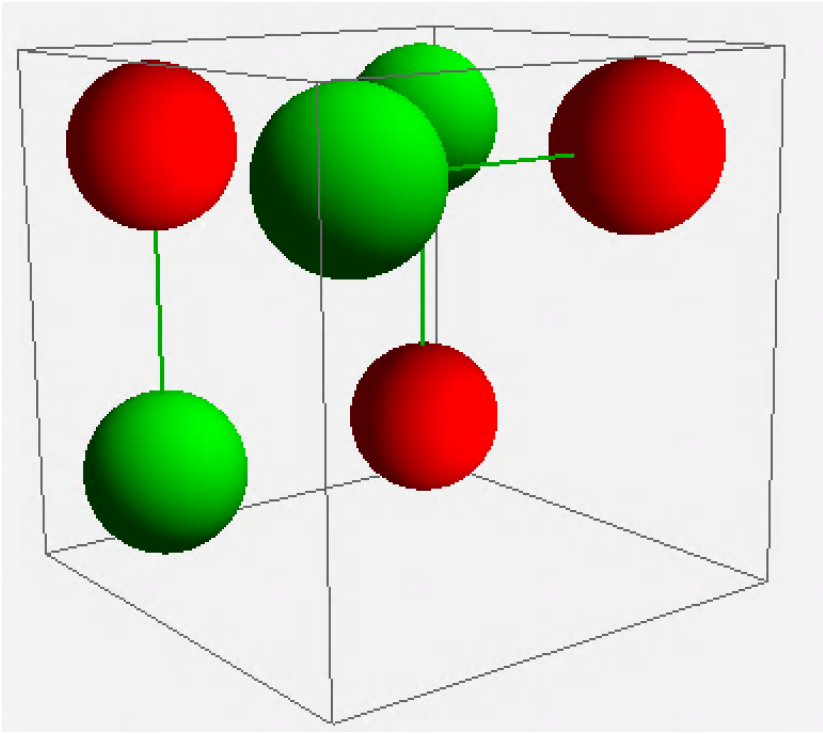


Figure 4.1.3: 3D Input (0 Contacts)

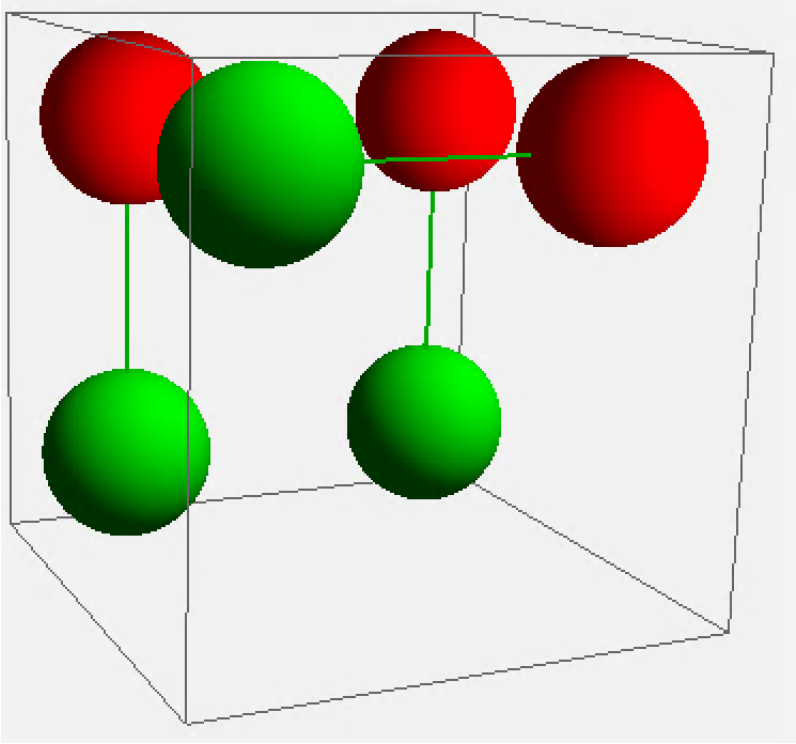


Figure 4.1.4: 3D Output (2 Contacts)

4.2 Explanation and Code

To tackle this problem, a backtracking technique was performed on the bisphere segments on the grid. More specifically, a program was created that recursively moves through the segments, flips them, and keeps track of which board configurations yielded the largest number of contacts. The logic is as follows: if the current bisphere segment is the last one remaining, update the current best solution and return to the previous recursive call. Otherwise, if the current flip is a valid option, the move is conducted, the resulting board is stored, and recursion continues with the next bisphere segment. The pseudocode to return a solution using backtracking is shown below. This pseudocode was made generic to enhance flexibility, but was primarily used for the bisphere packing problem. For this problem, the method takes in the current index corresponding to the segment being examined, flips the segment if appropriate, then continues. The algorithm must also have a reference to the total number of segments, to the flipping moves available, and to a solver that actually performs moves and validity checks.

Algorithm 1 Backtracking

```
1: procedure SOLVE(index, total, moves, solver)
2:     if index = total then
3:         solver.update_solutions()
4:         return
5:     for move in moves do
6:         if solver.move_is_valid(index, move) then
7:             solver.move(index, move)
8:             solve(index+1, total, moves, solver)
9:     return
```

Figure 4.2.1: Backtracking Pseudocode

4.3 Time Complexity and Modifications

Backtracking inherently has an exponential run time since every possible orientation of bisphere segments could, in theory, be examined. However, to increase the speed of the code, modifications were made that could cut down how much of the set of all possible solutions is searched. First, the initial board was turned into a "heuristic board," where squares of side chain circles were grouped together if possible. Additionally, a loop was done through the bisphere segments, and if flipping them resulted in a higher contact count, then they would be flipped. Using this heuristic approach, and by adding conditions that stop the program from continuing down the same search path if there is no possible way to get the optimal solution, the algorithm can be sped up significantly. It was difficult to generate conditions that decreased run time substantially while also not eliminating potential optimal solutions, but one solution was found. Namely, prior to making a move and proceeding to the next layer of the search process, the number of potential contacts is computed based on the number of segments remaining. If this value is less than either the current maximum or the maximum of the initial heuristic board, then there is no way for the present search path to yield an optimal solution. Thus, the program backtracks to a previous state. With this approach, the speed of the program was increased. However, the degree of the speed up varied based on the initial placement of bisphere segments on the board being used.

4.4 Results

The program output the desired configuration for both the 2D and 3D problems. In 2D, a problem consisting of around 35 bispheres could be solved in a few minutes, but after this point the running time was significant due to the exponential nature of backtracking. The maximum number of bispheres completed in 3D was similar, but slightly less due to the increased computation needed for another dimension. Some more complex inputs and outputs for 2D and 3D scenarios are shown below:

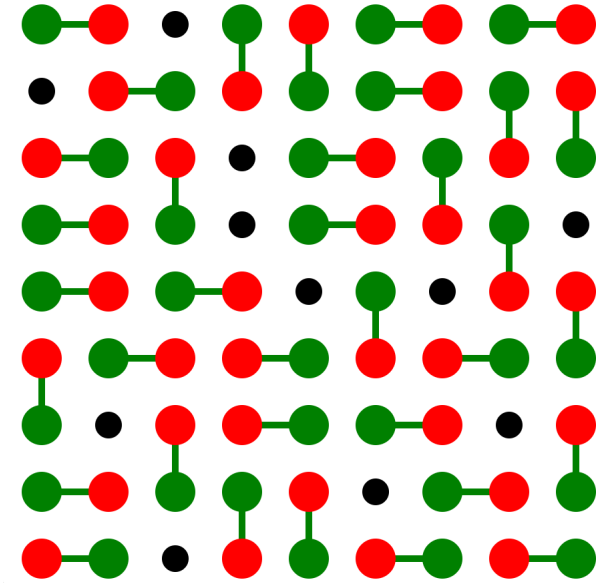


Figure 4.4.1: 2D Input (15 Contacts)

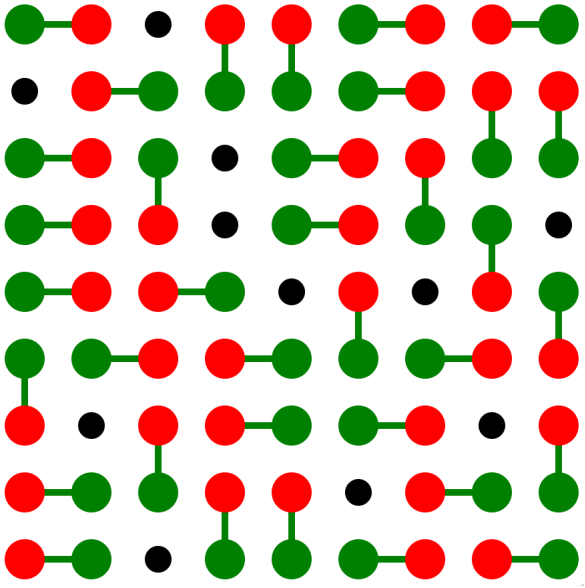


Figure 4.4.2: 2D Output (31 Contacts)

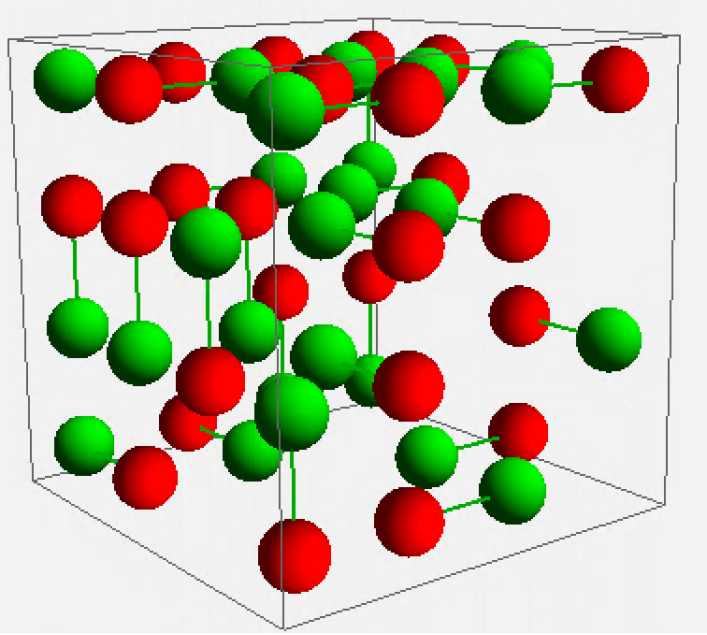


Figure 4.4.3: 3D Input (15 Contacts)

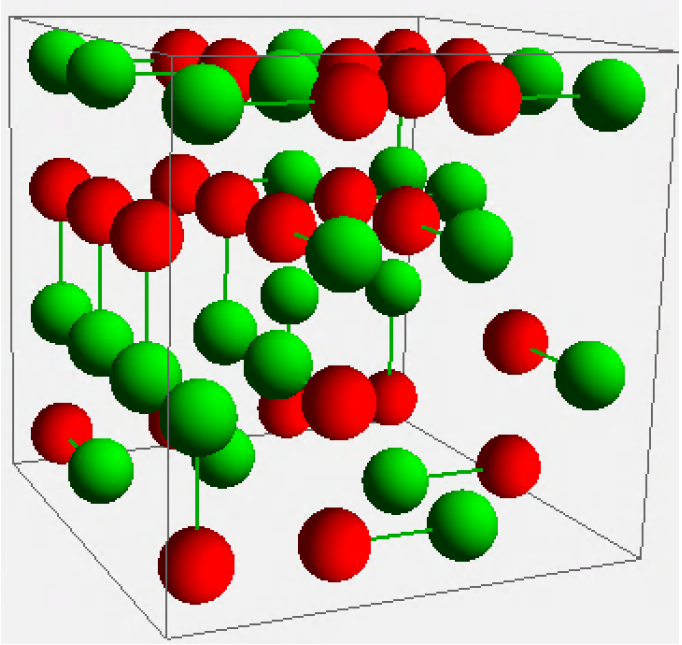


Figure 4.4.4: 3D Output (26 Contacts)

In 2D, a trend that was observed was that the segments tended to form a ladder structure whenever possible, as shown below.

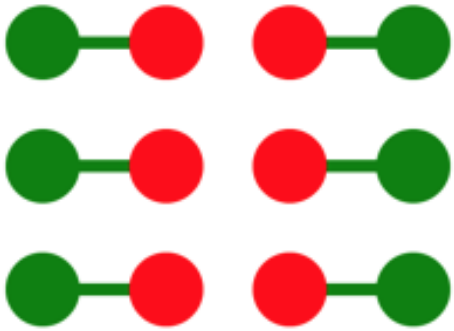


Figure 4.4.5: Ladder Structure

This configuration is reasonable for 2D packing, since this maximizes the number of contacts that any one bisphere segment can form. A similar trend was seen in 3D, except a bilayer structure was seen.

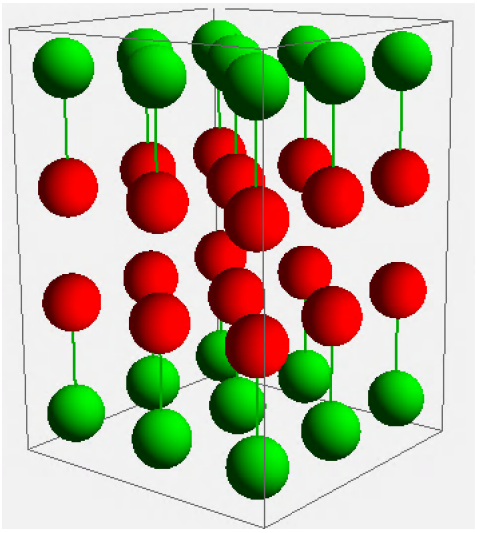


Figure 4.4.6: Bilayer Structure

Again, this pattern is reasonable since it packs the segments together in a way that is favorable for achieving a maximum number of contacts.

4.5 Future Work

There are many ways that the current backtracking implementation could be improved. In one of his papers, Donald Knuth speaks in detail about the "dancing links" data structure, which is a modified version of a doubly linked list. By using this data structure in the backtracking program, reverting to a previous state could be much more efficient and a lot of time could be saved [12]. Additionally, the code could be much faster if more conditions were added that could allow a search to end earlier without losing any optimal solutions. Generally, a large part of the total potential solution space is still searched, which takes a long time. With more conditions and a new data structure, the backtracking algorithm could run much faster.

5 SAT Solver Optimization

5.1 SAT Solvers

A SAT solver is a tool that can be used to solve problems in the following format: given some set of variables with a series of constraints, output the values of these variables that satisfy all the constraints. Since the protein folding problems being studied deal with maximizing contacts, a traditional solver was not used, but rather an optimizer that maximizes some value given a set of variables with constraints. For the purposes of this project, a Z3 optimizer was used to generate an output for the problem.

5.2 Representation and Problem

As in the backtracking problem, bisphere segments are the primary units that will be used. However, the problem that is being solved with the SAT optimizer is different than the one tackled with backtracking. Instead, the problem is as follows: given some 2D rectangular lattice of dimensions x by y, how can n bisphere segments be placed such that the number of contacts is maximized? This is different from the prior situation because the necessary decision is not how to flip the bisphere segments, but rather how to place them in the first place.

5.3 Explanation and Code

To force the SAT optimizer to generate the desired solutions, numerous kinds of variables and constraints needed to be created. Firstly, 2D matrices of boolean variables corresponding to backbone spheres and side chain spheres were made, as well as a boolean array corresponding to the edges between these spheres. If a value in one of these arrays is true, this indicates that a sphere or edge is present at the given index. Constraining these arrays allows the program to limit where spheres and edges can be placed, which in turn grants the ability to construct a legal board. Once the ground rules for how bispheres must be placed were set, the optimizer is told to maximize the number of times a side chain sphere is next to another side chain sphere. In other words, the number of contacts will be maximized, and the configuration that gives this maximum value will be output. The Python code for creating some constraints is shown below:

```
# each spot can only have one of the following: backbone, sidechain, empty

for i in range(y):
    for j in range(x):
        # cannot have a backbone and a sidechain at index (i, j)
        constraints_spheres.append((Not(And(backbone[i][j], sidechain[i][j]))))
```

Figure 5.3.1: Constraint: Only One Bisphere Endpoint per Grid Entry

```
# must have a number of backbone spheres and sidechain spheres equal to the number of bispheres

# create integer matrices that will hold either a 1 or 0
backbone_spheres = [Int("bb_%s" % (i+1)) for i in range(area)]
sidechain_spheres = [Int("sc_%s" % (i+1)) for i in range(area)]

count = 0

for i in range(y):
    for j in range(x):
        backbone_spheres[count] = If(backbone[i][j], 1, 0) # 1 if backbone present, 0 if not
        sidechain_spheres[count] = If(sidechain[i][j], 1, 0) # 1 if sidechain present, 0 if not
        count += 1

# number of 1s on board for backbone and sidechain must equal number of bispheres
constraints_num_spheres.append((Sum(backbone_spheres) == num_bispheres))
constraints_num_spheres.append((Sum(sidechain_spheres) == num_bispheres))
```

Figure 5.3.2: Constraint: Number of Backbone and Side Chain Endpoints Must Equal Number of Bisphere Segments

```
# number of edges must be equal to the number of bispheres

# create integer array that will hold either a 1 or 0
edge_values = [Int("ee_%s" % (i+1)) for i in range(edges_possible)]

for i in range(edges_possible):
    edge_values[i] = If(edges[i], 1, 0) # 1 if edge present, 0 if not

# number of 1s in edge array must equal number of bispheres
constraints_num_edges.append((Sum(edge_values) == num_bispheres))
```

Figure 5.3.3: Constraint: Number of Edges Must Equal Number of Bisphere Segments

5.4 Time Complexity and Symmetry Breaking

Due to the brute force nature of a SAT solver, as well as the fact that the number of constraints required grows exponentially as the number of bisphere segments increases, the program has an exponential run time which causes it to take a very long time for values over 10 bisphere segments. To combat this issue, horizontal and vertical versions of lex-leader symmetry breaking were implemented. In other words, the solver will view some configuration and its reflection across the x or y axis the same. Other approaches to cutting down the run time of the SAT solver involved fixing the position of a bisphere segment in the first row or column of the board, with the goal of stopping the solver from generating

the same configuration on different parts of the board. Unfortunately, this approach, as well as the symmetry breaking attempts, did not conclusively improve the run time of the solver.

5.5 Results

The program outputs the desired bisphere segment placement on the 2D rectangular lattice. As observed with the backtracking method, the bisphere segments tend to form a 2D ladder structure given they have enough space on the grid. The program finishes in a reasonable amount of time for values up to 10 bisphere segments, but at this point the run time increases drastically. An optimal arrangement for 9 bisphere segments that was generated by the solver is shown below:

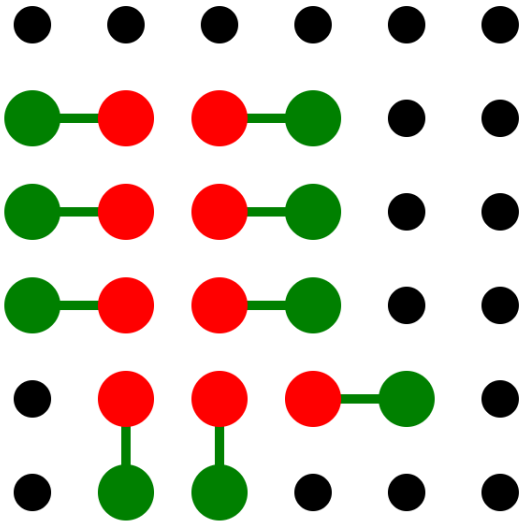


Figure 5.5.1: Z3 Optimizer Output for 9 Bisphere Segments on a 6 by 6 Grid

5.6 Future Work

The SAT solver optimization approach that was taken is still very flawed and can likely be improved in several ways. Firstly, finding a more efficient way to express the constraints required for the problem could improve run time significantly. Also, searching for a more effective form of symmetry breaking could reduce the amount of work the solver must do by a large amount. Aside from this, using a different optimizer other than Z3, as well as running the code on a more powerful computer, could make the program faster. Alternative methods, such as backtracking, should also be explored in the context of this problem to see if they perform better than brute force SAT solver optimization. Once improvements are made to the 2D version of the optimization, it should be expanded to 3D.

6 Conclusion

The intricacies of protein folding are still not well understood, despite enormous efforts from both the computer science and biology communities. A better understanding of what drives this process is slowly being acquired, with the hope of one day being able to predict how a protein will fold using only on its amino acid sequence. This would be a groundbreaking achievement for many reasons, including the potential for new treatments in the realm of protein misfolding diseases such as Alzheimer's and prion diseases. For this Honors Thesis, an exploration was made in the complex and exciting field of simplified protein folding prediction. Namely, a PDB file parser and a visualizer were created to improve analysis of how hydrophobic interactions impact folding patterns. Additionally, algorithms and optimization approaches were developed for a variant of the HP lattice model of proteins, where backbone-side chain pairs are represented as segments with circles on both ends. Through continued examination of hydrophobic interactions on both actual and simplified models, significant progress could be made in the study of protein folding prediction.

References

[1] O. Šerý, J. Povová, I. Míšek, L. Pešák, and V. Janout, Molecular mechanisms of neuropathological changes in Alzheimer's disease: a review, Folia Neuropathologica, vol. 1, pp. 1–9, 2013.

[2] M. D. Geschwind, Prion Diseases, CONTINUUM: Lifelong Learning in Neurology, vol. 21, pp. 1612–1638, 2015.

[3] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. J. Comp Bio, 5:27–40, 1998.

[4] R. Backofen. The protein structure prediction problem: A constraint optimization approach using a new lower bound. Constraints, 6:223–255, 2001.

[5] V. Chandru, A. DattaSharma, and V. S. Anil Kumar. The algorithmics of folding proteins on lattices. Discrete Applied Mathematics, 127:145–161, 2003.

[6] K. A. Dill. Theory for the folding and stability of globular proteins. Biochemistry, 24, 1985.

[7] K. A. Dill. Dominant forces in protein folding. Biochemistry, 29:7133–7155, 1990.

[8] W. E. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. Journal of Computational Biology, 4:1–22, 1997.

[9] D. Shaw, A. S. Islam, M. S. Rahman, and M. Hasan, Protein folding in HP model on hexagonal lattices with diagonals,BMC Bioinformatics, vol. 15, no. Suppl 2, Jan. 2014.

[10] Q. Guo, J. Wang, and Z. Xu, Approximation Algorithms for Protein Folding in the Hydrophobic-Polar Model on 3D Hexagonal Prism Lattice, Journal of Computational Biology, vol. 25, no. 5, pp. 487–498, 2018.

[11] S. Istrail and F. Lam, "Combinatorial Algorithms for Protein Folding in Lattice Models: A Survey of Mathematical Results,"Communications in Information and Systems, vol. 9, no. 4, pp. 303–346, 2009.

[12] D. E. Knuth, "Dancing Links,"arXiv:cs.DS/0011047, Nov. 2000.

Figures