University of Massachusetts Dartmouth

Department of Computer and Information Science

MDS (Multi-Dimensional Scaling) visualization for Foldit Results

A Thesis in

Computer Science

By

Anand Shah

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Spring 2016

I grant the University of Massachusetts Dartmouth the non-exclusive right to use the

work for the purpose of making single copies of the work available to the public on a not-

for-profit basis if the University's circulating copy is lost or destroyed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Anand Shah

Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

We approve the thesis of Anand Shah

Date of Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Firas Khatib  
Assistant Professor of Computer and Information Science

Thesis Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. David Koop  
Assistant Professor of Computer and Information Science

Thesis Committee

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Shelley Zhang  
Associate Professor of Computer and Information Science

Thesis Committee

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Dr. Jan Bergandy  
Chairperson, Department of Computer and Information Science

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Robert E. Peck  
Dean, College of Engineering

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Tesfay Meressi  
Associate Provost for Graduate Studies

**ABSTRACT**

**Title:  MDS(Multi-Dimensional Scaling) visualization for Foldit results**

By Anand Shah.

Foldit is a protein folding video game where players try to find the correct folds of different proteins. The players are given a protein structure as a puzzle to solve and attempt to maximize their in-game score by folding the protein in the best possible way. Foldit players are able to form teams and share their solutions with one another, competing against other Foldit teams.

This thesis aims to help the players see how they performed compared to other teams on a given puzzle. Our approach is to use Multi-Dimensional Scaling (MDS) to compare the different Foldit solutions to each other. This is done by performing a pairwise comparison of the top scoring player solutions, which is then fed to an MDS algorithm. The results are then plotted on a multi-dimensional graph showing how similar/different they are from each other, along with the Foldit scores of each player solution. The protein structures can also be viewed in a protein viewer by clicking on individual points of the MDS graph. The protein viewer shows how exactly the protein structures are different from each other. One of the advantages of this visualization is the ability to analyze the results of a puzzle very easily. A quick glance at the graph is sufficient to get an idea about how well the players performed on a particular puzzle. The goal of this project is to provide this feedback to the Foldit players so that they can improve their solutions on future puzzles.

**ACKNOWLEDGEMENT**

Firstly, I would like to thank my thesis advisor Dr. Firas Khatib for his support and help during my Master’s program as well as my thesis. I would like to thank him for introducing me to Bioinformatics and lots of cool stuff that came along my way in the form my coursework, internship, conferences, project and thesis. Sincere thanks for letting me be a part of RosettaCons, Cyrus biotech and Rosetta. I would like to thank him for letting me be his Research Assistant. It was an honor working on Foldit and Rosetta. It was fun and educational doing hours of research on them. Dr. Khatib helped me to be on track with my work through weekly meetings. He always kept pushing me as a result of which I could complete my project and thesis on time. He was also quick to respond on queries and I always found help from him and Foldit/Rosetta team when I needed it the most. Dr. Khatib is one of the most intelligent and the best advisor that one can wish for. I cannot thank him enough for everything.

I would like to thank Dr. Xu for his valuable feedback during my project defense, which helped me learn more about MDS. I would like to thank Dr. Koop, the visualization expert, for helping me refine my MDS visualization. I would like to thank Dr. Zhang for her valuable feedback during my thesis. I would like to thank Mahmood for his valuable feedback during project rehearsals. I would like to thank Shashank, Nick and Main for their awesome support.

I would sincerely like to thank our Foldit team members Jeff, Seth, Brian, Steven Combs, Nova, Vikram for their valuable inputs and feedback on MDS. Special thanks to Jeff for always being on top of things and getting MDS up and running on the Foldit website. I would also like to thank Seth for sending me links to crucial stuff that I needed the most for my project and thesis like MDS jar file which is the backbone for the entire MDS calculations and also research papers which I could relate to MDS. I was happy to see Brian and other Foldit team members using MDS for analysis. I would like to thank Dr. Khatib, Jeff, Seth and Brian for our discussions during Foldit meetings and RosettaCons. Being at RosettaCon was truly fun and educational. I would like to thank Dr. Khatib to let me present MDS at RosettaCon which helped me get my MDS concepts clearer by answering to people’s questions. Special thanks to Dr. Roland Dunbrack and his team Vivek and Jared for helping me on MDS. I would also like to thank Dr. David Baker for being the pioneer of Rosetta and RosettaCons. Lastly I would like to thank all my RosettaCon friends for their valuable feedback.

I would like to thank my Cyrus Biotech mentors Javier, Lucas and Yifan for letting me intern for their company in Seattle. I learnt a lot from them during my stay and I hope to use that knowledge in the future. Their feedback on MDS was also very useful.

Finally I would like to thank my Family for being the source my strength. I would like to thank my friends for their continuous support and also for being there during my project and thesis defense.

TABLE OF CONTENTS

[LIST OF FIGURES…….………………………………..……….....vi](#_Toc438926015)

[1. Introduction……... 1](#_Toc438926016)

[1.1 What are proteins? 1](#_Toc438926017)

[1.2 What is protein folding? 2](#_Toc438926018)

[1.3 Native state of a protein structure 3](#_Toc438926019)

[1.4 Energy of a protein structure 4](#_Toc438926020)

[1.5 What is a PDB? 4](#_Toc438926021)

[1.6 What is Foldit :- 5](#_Toc438926022)

[1.7 Why is Foldit important? 6](#_Toc438926023)

[1.8 What is Rosetta? 6](#_Toc438926024)

[1.9 Motivation for the thesis 7](#_Toc438926026)

[2. Background Knowledge 9](#_Toc438926027)

[2.1 RMSD 9](#_Toc438926028)

[2.2 GDT\_TS 9](#_Toc438926029)

[2.3 TMAlign 10](#_Toc438926030)

[2.4 MDS 11](#_Toc438926031)

[2.5 Use of MDS in Foldit/Rosetta 13](#_Toc438926032)

[2.6 More about PDBs 14](#_Toc438926033)

[2.7 D3 - The visualization tool 16](#_Toc438926034)

[2.7.1 Introduction to D3 (Data Driven Documents) 16](#_Toc438926035)

[2.7.2 Use of D3 to analyze Foldit results 16](#_Toc438926036)

[2.7.3 Data file for Force Directed Graph 18](#_Toc438926037)

[2.7.4 Data file for MDS 19](#_Toc438926038)

[3. Related work…….. 21](#_Toc438926039)

[3.1 GDT\_TS vs Energy plot 21](#_Toc438926040)

[3.2 Previous analysis 24](#_Toc438926041)

[3.3 Why it did not work(Drawbacks) 27](#_Toc438926043)

[3.4 Feature-Based Projections for Effective Playtrace Analysis 28](#_Toc438926044)

[4. Methods & Results ……………………………………………….30](#_Toc438926045)

[4.1 Python script 30](#_Toc438926046)

[4.2 MDS on the Foldit website 32](#_Toc438926047)

[4.3 PV in MDS 43](#_Toc438926048)

[4.4 Data file for MDS visualization 44](#_Toc438926049)

[5. Conclusion & Future work 46](#_Toc438926054)

[References 47](#_Toc438926055)

# LIST OF FIGURES

[Figure 1: Example of a complex protein structure. 1](#_Toc438839691)

[Figure 2 : Example of a simple protein structure. 2](#_Toc438839692)

[Figure 3: Example of Protein folding. 3](#_Toc438839693)

[Figure 4: Native state of a protein. 4](#_Toc438839694)

[Figure 5: Protein folding in Foldit. 5](#_Toc438839695)

[Figure 6: RMSD 9](#_Toc438839696)

[Figure 7: Two dimensional matrix for distance between cities. 11](#_Toc438839697)

[Figure 8: MDS plot for distances between cities. 12](#_Toc438839698)

[Figure 9: Stress function for MDS 13](#_Toc438839699)

[Figure 10: PDB structure 15](#_Toc438839700)

[Figure 11: Example of a Force directed graph 16](#_Toc438839701)

[Figure 12: Example of a MDS plot 17](#_Toc438839702)

[Figure 13: Sample data file format of a Force Directed Graph 18](#_Toc438839703)

[Figure 14(A): Sample data file for MDS(x-y coordinates) 19](#_Toc438839704)

[Figure 14(B): Sample data file for MDS(Two-dimensional matrix) 20](#_Toc438839705)

[Figure 14(C): Sample data file for MDS(City names) 20](#_Toc438839706)

[Figure 15: GDT\_TS plot 21](#_Toc438839707)

[Figure 16: RMSD plot 22](#_Toc438839708)

[Figure 17(A): All group selection for previous visualization (Force Directed Graph) 24](#_Toc438839709)

[Figure 17(B): Single group selection for previous visualization (Force Directed Graph) 25](#_Toc438839710)

[Figure 17(C): Different datasets for previous visualization. 26](#_Toc438839711)

[Figure 18: Output of playtracer on a puzzle game. 28](#_Toc438839712)

[Figure 19: Playtracer trajectory for Foldit winners and losers. 29](#_Toc438839713)

[Figure 20: Puzzle section on the the Foldit website 32](#_Toc438839714)

[Figure 21: A closed puzzle on the Foldit website 33](#_Toc438839715)

[Figure 22: GDT\_TS, RMSD & MDS plots for a puzzle. 33](#_Toc438839716)

[Figure 23: MDS plot for a Foldit puzzle. 34](#_Toc438839717)

[Figure 24: Selecting-deselecting of teams in MDS 36](#_Toc438839718)

[Figure 25: Start, Native and PDB file name on the MDS plot 37](#_Toc438839719)

[Figure 26: Top 10 teams in the MDS plot 38](#_Toc438839720)

[Figure 27: PDBs with the best energies on the MDS plot. 39](#_Toc438839721)

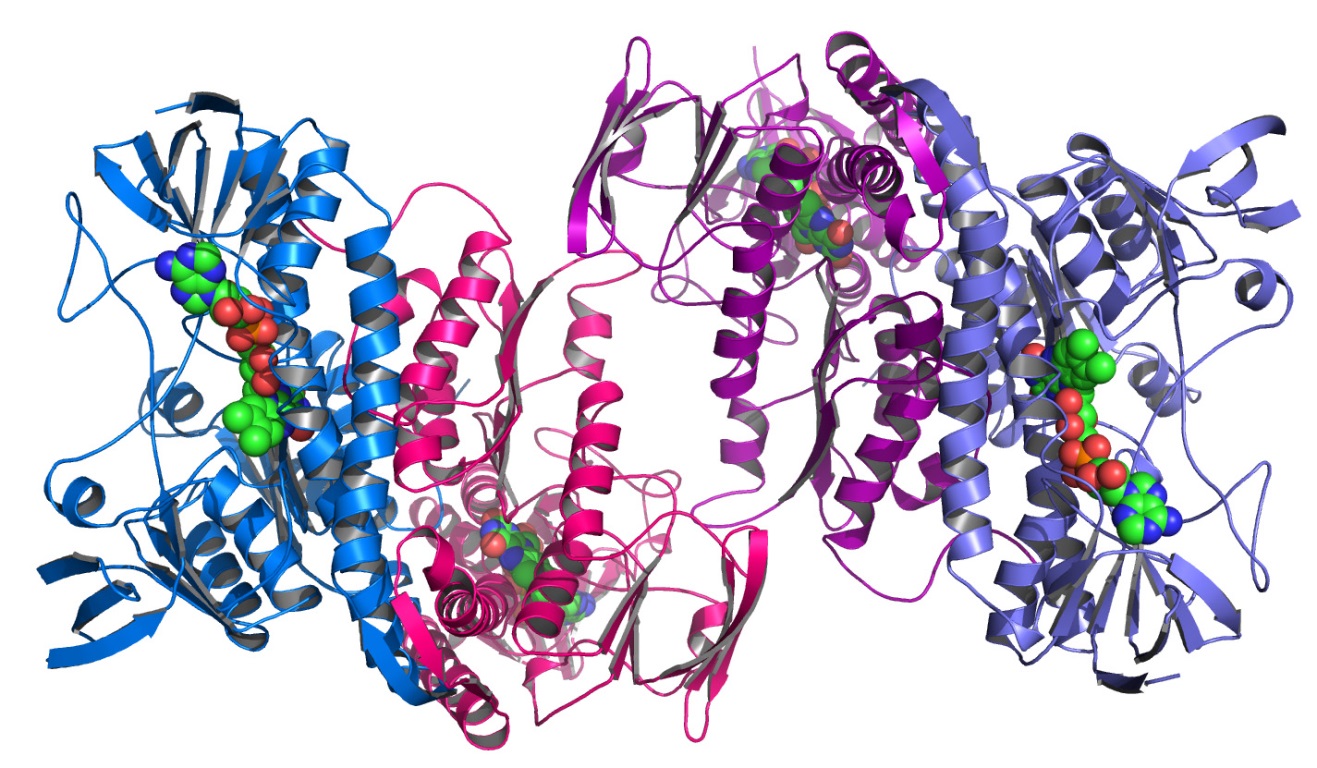
[Figure 28: PV protein viewer in MDS. 41](#_Toc438839722)

[Figure 29: Data for a PDB dot in the MDS data file. 44](#_Toc438839723)

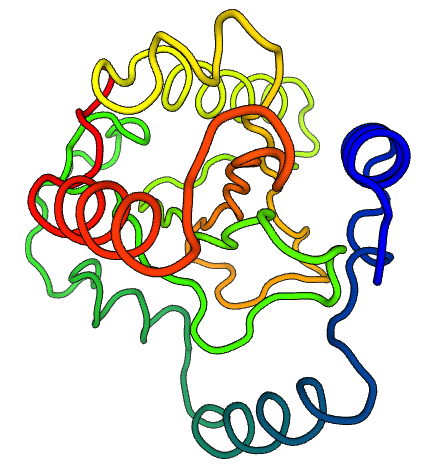
# Introduction

## 1.1 What are proteins?

Proteins are building blocks of life. For a human body, proteins are essential for the growth and repair of bones and muscles. Proteins help to make tissue and cells. When proteins are digested into the body, they are broken down into amino acids. So proteins are long-chain molecules that are built from small blocks of **amino acids**. Proteins are essential to all the cells, that is, they take part in almost every process in cells. Some proteins help to build muscles and bones. Some proteins are important in cell signaling, immune responses and cell division.



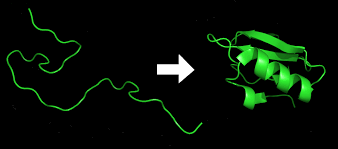
**Figure 1: Example of a complex protein structure.**



**Figure 2 : Example of a simple protein structure.**

## 1.2 What is protein folding?

Even though proteins are a long chain of amino acids, they usually don’t stay stretched out in a straight line. The long chain of amino acid folds up into its best shape such that it is stable and has the lowest free energy. The structure of a protein usually tells us its function. For example, a protein that breaks down glucose so that the cell can use the energy stored in the sugar will have a shape that recognizes the glucose and binds to it like a jigsaw puzzle.

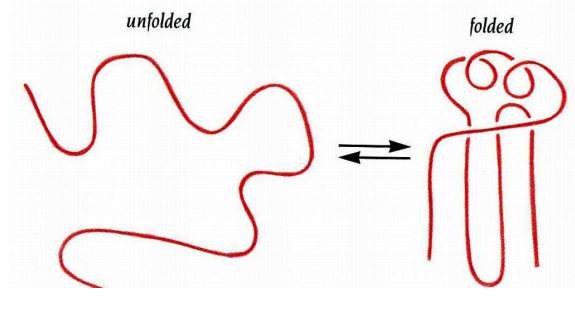


**Figure 3: Example of Protein folding.**

In the above figure, the stretched out chain of amino acids are folded in such a way that the folded protein structure is in its most stable state and has the lowest possible energy.

## 1.3 Native state of a protein structure

Native state of a protein structure is achieved by folding a protein structure in such a way that the protein structure is unique, stable and has the lowest energy. **Unique** means that the sequence does not have any other fold with a lower energy. **Stable** means that the energy of the protein structure cannot change because of environmental factors.[[1]](#endnote-1)



**Figure 4: Native state of a protein.**

## 1.4 Energy of a protein structure

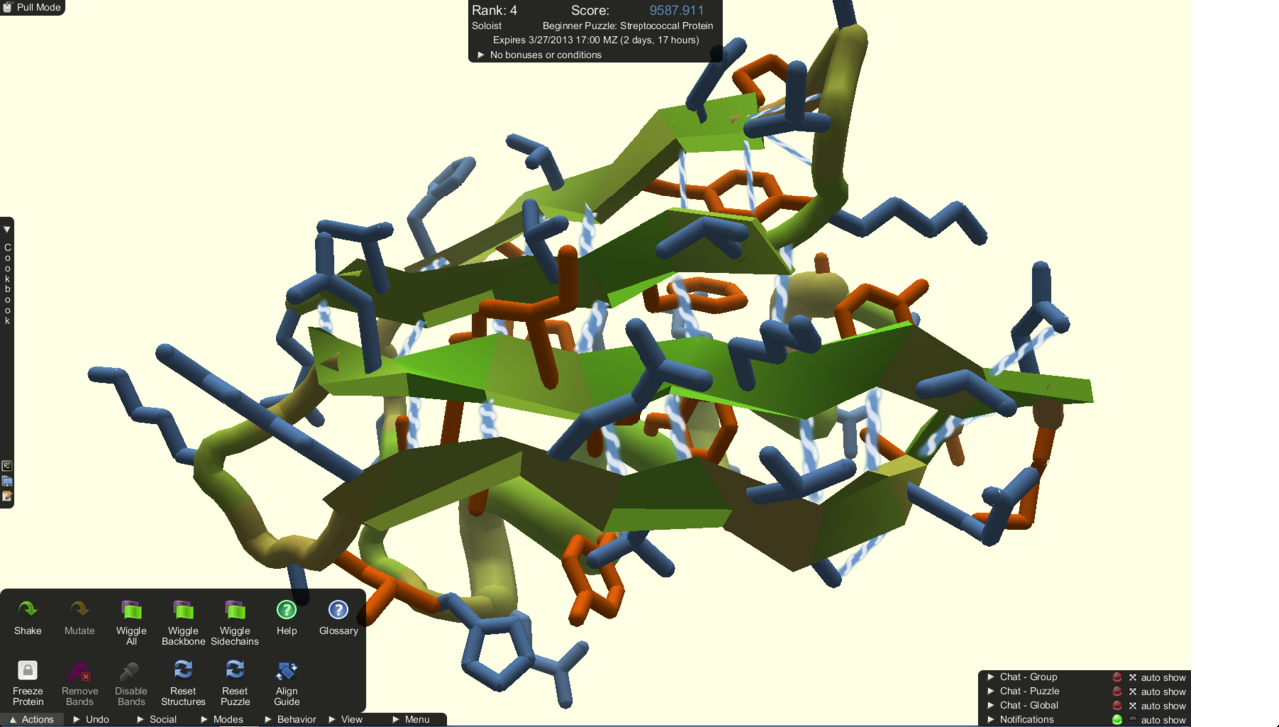
Every protein structure is associated with an energy. This energy has to be as low (negative) as possible for the protein to be in its native state.

# 1.5 What is a PDB?

The Protein Data Bank (PDB) file is a file structure that holds the x, y and z coordinates of a protein structure along with the secondary structure assignments and other atom information.

## 

## 1.6 What is Foldit?

****

**Figure 5: Protein folding in Foldit.**

Foldit is an online protein folding video game. It is part of an experimental research project at the University of Washington. The players are given a protein structure as a puzzle to solve and they try to maximize the score by folding the protein in the best possible way, thereby trying to get their structure as close to the native as possible and sometimes even better than the native. The Foldit players form teams and they share their solution within the team. They also compete with other Foldit teams.[[2]](#endnote-2)

## 1.7 Why is Foldit important?

Foldit tries to predict the structure of a protein by using a human’s puzzle-solving instincts and making people compete against each other to find the best possible protein structure. Players can also design new protein structures that could cure important diseases. The players will know more about protein folds if they keep on playing the game. The more they know about protein folds, the better they will do on designing new proteins and cure diseases. Some of the most important diseases for which Foldit is trying to find a cure are HIV/AIDS, Cancer, Alzheimer’s and Ebola. The cure for these diseases would be revolutionary. We can use Foldit not only for human proteins but for other organisms as well. By learning how humans play the game, we can train the machines to do the same and then we would be able to fold proteins faster than ever.

## 

## 1.8 What is Rosetta?

Rosetta is an algorithm that automatically accomplishes prediction, design and analysis of the protein structures. Rosetta accomplishes protein folding automatically whereas Foldit does it manually. All the tools in Foldit are simplified versions of the Rosetta algorithm. Foldit is developed to help Rosetta get better. If the Foldit players are able to find a better solution than Rosetta, that means Rosetta has a loophole that needs to be improved.

## 

## 1.9 Motivation for the thesis

In Foldit, the players need to compare their solutions with other teams. They need to see how they performed in a puzzle. The only way to do this is to visualize the structural distances between the protein structures. A visualization is needed to show the distances at a glance. The Foldit website has a result section that shows the GDT\_TS vs Energy and RMSD vs Energy plots. These plots compare all the PDBs (Protein data bank files) to the native but do not compare the PDBs to each other. So we needed a visualization that would compare all the PDBs to each other.

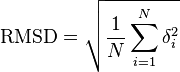
This thesis presents a tool that shows how similar/different the protein structures are from each other using a MDS (Multi-Dimensional Scaling) plot which is discussed in section 2. This tool can be used to analyze Foldit/Rosetta results.

The rest of the thesis is organized as follows. Section 2 talks about background knowledge to help fully understand our approach. Section 3 talks about related work. Section 4 describes the methods and results. Finally, section 5 summarizes our conclusion and discusses the future work.

# Background Knowledge

## 2.1 RMSD

RMSD is one of the measures of similarity between protein structures. Protein structures are superimposed and the distance between all the atoms is calculated. The average of all the sum of the distances gives us the RMSD. [[3]](#endnote-3),[[4]](#endnote-4),[[5]](#endnote-5),[[6]](#endnote-6),[[7]](#endnote-7),[[8]](#endnote-8)



**Figure 6: RMSD**

In the above figure, δ is the distance between N pairs of equivalent atoms.

## 

## 2.2 GDT\_TS

GDT\_TS is another way to measure similarity between protein structures. Protein structures are superimposed and their similarity is decided on the basis of an Angstrom distance cutoff between the atoms of the proteins. Similarly, the protein structures are compared based on different cutoffs and the average of all the distances gives us the GDT\_TS measure.

GDT\_TS= 1/4(maxC1Å+maxC2Å+maxC4Å+maxC8Å)

Where C1Å is the number of atom pairs below the 1Å distance. Max denotes here the maximum value for a series of superimpositions. [[9]](#endnote-9),[[10]](#endnote-10),[[11]](#endnote-11)

GDT\_TS is more accurate than RMSD because RMSD is just an average between sums of the distances between atoms whereas GDT\_TS is based on angstrom cutoffs and the average of all the cutoffs. So imagine a scenario in which two protein structures are exactly the same except for their tail residues, which are totally different. In this case, RMSD would not be as accurate as GDT\_TS. The RMSD measure would show a 60% similarity whereas GDT\_TS would show 90% similarity between proteins because of the angstrom cutoffs (The percentages mentioned above are approximate measures.)

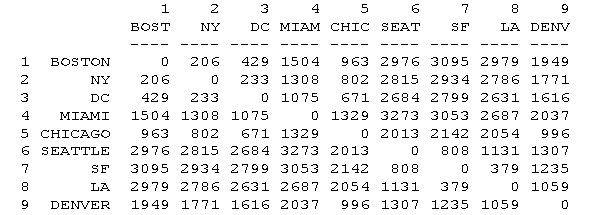
## 

## 2.3 TMAlign

TMAlign is a software that calculates the similarity between proteins. TMAlign generates TMScore, which is the measure of similarity between two proteins. TMAlign is better than GDT\_TS as it adds one more factor to the similarity, that is, the length of the protein. Thus TMScore is the best measure we have. A TMScore of 1 means that the protein structures are exactly the same and a TMScore of 0 means that the protein structures are totally different. [[12]](#endnote-12),[[13]](#endnote-13),[[14]](#endnote-14)

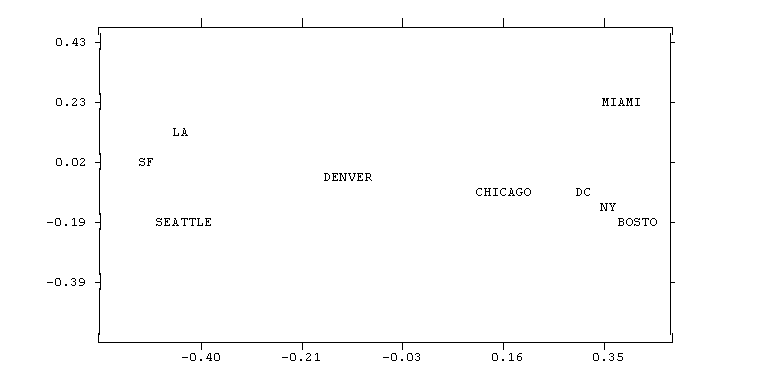
## 2.4 MDS

MDS (Multidimensional Scaling) is a method to provide a visual representation of the pattern of proximities (i.e., similarities or distances) among a set of objects. MDS is a visual representation of distances between a set of points in the form of a scatter plot. Consider the below example in which we need to represent distances between cities.[[15]](#endnote-15)

****

**Figure 7: Two dimensional matrix for distance between cities.**

We feed the 2 dimensional matrix to the MDS algorithm which outputs us the coordinates to be plotted on the scatter plot.

****

**Figure 8: MDS plot for distances between cities.**

Above is the plot that gives approximate distances between cities. The numbers on the x-axes mean the x coordinate and the numbers on the y-axes mean the y coordinate of the cities plotted on the graph. Using the x and y co-ordinates, we can use the distance formula to find the distance between the cities. The distance that we get is approximately close to the distances in the two-dimensional distance matrix.

Explaining the concept further, MDS represents the distance between objects as distances between objects of a low-dimensional space. An MDS algorithm tries to place each object in an n-dimensional space such that the distance between objects is as accurate as possible. Each object is assigned coordinates in each of the n dimensions. Choosing 2 dimensions for MDS optimizes the location for the objects in a 2 dimensional scatter plot.

MDS finds a set of coordinates in some dimensional space such that the Euclidean distances between the objects correspond as closely as possible to some function of the input matrix according to a function called *Stress*.



**Figure 9: Stress function for MDS**

In the above equation, dij refers to the euclidean distance across all dimensions between points i and j on the map, f(xij) is some function of the input data, and scale refers to a constant scaling factor, used to keep stress values between 0 and 1. When the MDS map perfectly reproduces the input data, f(xij) - dij is 0, so the stress is zero. Thus the smaller the stress, the better the representation. [[16]](#endnote-16)

## 2.5 Use of MDS in Foldit/Rosetta

The purpose of MDS is to compare PDBs to each other and see how similar/different they are from each other. For a Foldit team, the more diverse their solutions, the better are their chances of finding the right solution. We try to see which group has the most diverse solutions. By showing the plots to the Foldit players, we encourage them to diversify their solutions. In a group, if the solutions of all the players are diverse, the chances of their solution being close to the native are more.

This tool will be useful for the Foldit developers and the players. It will be useful for the Foldit players to compare their solutions to each other. The developers will be able to see how the players performed in the puzzle just by looking at the MDS plots at a glance.

In case of Rosetta, we can simply see how different the Rosetta solutions are from each other.

MDS in our case would show how different protein structures are from each other. Proteins data bank (PDB) files are plotted as a dot on the MDS plot. The distances between the PDBs are their TMScore values that are calculated using TMAlign. When we do a pairwise comparison and get the TMScore value, we invert the value and create an inverted 2-dimensional TMScore matrix. The reason to invert the TMScore values is that if we need to show that the PDBs are exactly the same, the distance between them should be 0 but their TMScore value is actually 1. So we need to invert every TMScore value in order to visually show the distances between them.

## 2.6 More about PDBs (Protein Data Bank files)

The Protein Data Bank(PDB) file is a file structure that holds the x, y and z coordinates of a protein structure along with the secondary structure assignments and other atom information. If we know the x, y and z coordinates of a protein structure we can view it in a protein structure visualizer and we can also compare it with other protein structures by superimposing them three dimensionally. Below is an example of a PDB file. The highlighted section has the x, y and z coordinates of a protein structure.

****

**Figure 10: PDB structure**

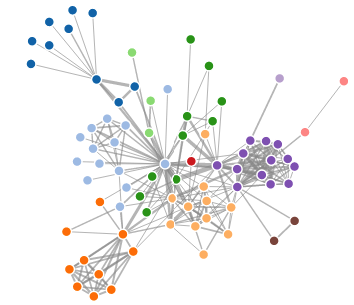
In Foldit, when the players submit a solution, they submit it in the form of a PDB. So at the end of the puzzle, we get PDBs from every player who played the puzzle. Rosetta also generates PDBs.

## 2.7 D3 - The visualization tool

### 2.7.1 Introduction to D3 (Data Driven Documents)

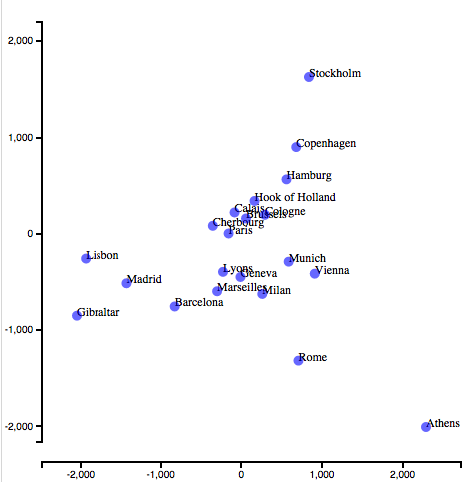
D3.js is a JavaScript library for creating interactive HTML pages. D3 helps us create great animations with not so complicated code. D3 is compatible with all the modern browsers. D3 can be used to create smooth transitions and interactions. D3 is used in combination with HTML, SVG and CSS. D3 is very fast, supports large datasets and supports dynamic behavior for interaction and animation. D3 is easy to debug using the browser’s built-in element inspector.[[17]](#endnote-17)

### 2.7.2 Use of D3 to analyze Foldit results



**Figure 11: Example of a Force directed graph**

Using D3’s Force Directed Graph, we tried to show the structural difference between the proteins. Above is an example of a force directed graph. Depending on how similar/different the protein structures are from each other, we can draw a line between the circles. The length of the line will tell us how different the protein structures are from each other. For example, if the protein structures are similar, the length of the line between the circles will be small whereas, if the protein structures are totally different, the length will be more. This was the first attempt at showing how similar/different the protein structures are from each other.



**Figure 12: Example of a MDS plot**

The above plot is an example of MDS using D3. In the above plot, the distance between cities is shown using a scatter plot type of graph.

In our case, the dots would be the protein structures and we can measure the distance between them using the axes. Details about MDS are explained in the “Background knowledge” section.

### 

### 2.7.3 Data file for Force Directed Graph

As D3 is data driven, all the visualizations require a data file that would have the data needed to create the visualization. The format of the data file is always specific to the type of visualization. For example, a force directed graph consists of nodes and edges, so the format of the data file will be in the form of nodes and links (edges).





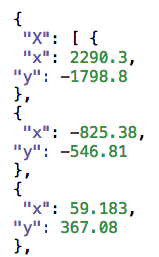
**Figure 13: Sample data file format of a Force Directed Graph**

In the above figure, the “nodes” has the name of the node and the group it belongs to. The other information that the data file would have is the “links”, meaning the information needed to draw edges between the circles. The “links” section would have the source node of the edge, destination node of the edge and the length of the edge.

### 

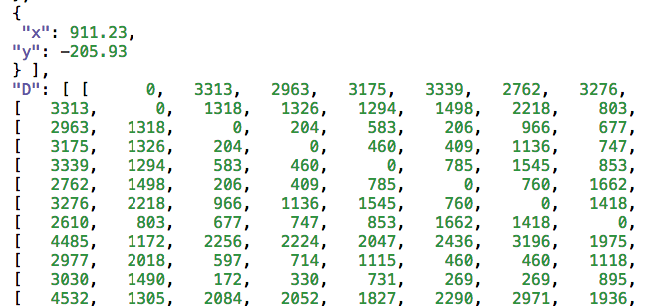
### 2.7.4 Data file for MDS

The MDS data file needs the X and Y coordinates of the dot, the two-dimensional distance matrix and the name of the dot. So accordingly, the data file would have all the above information.



**Figure 14(A): Sample data file for MDS(x-y coordinates)**

In the above figure, the “X” section has the x & y coordinates of the dot.



**Figure 14(B): Sample data file for MDS(Two-dimensional matrix)**

After the “X” section, the “D” section would have the two-dimensional matrix containing the actual distances between the dots.



**Figure 14(C): Sample data file for MDS(City names)**

Finally, the “cities” section would have the name of the cities (dots).

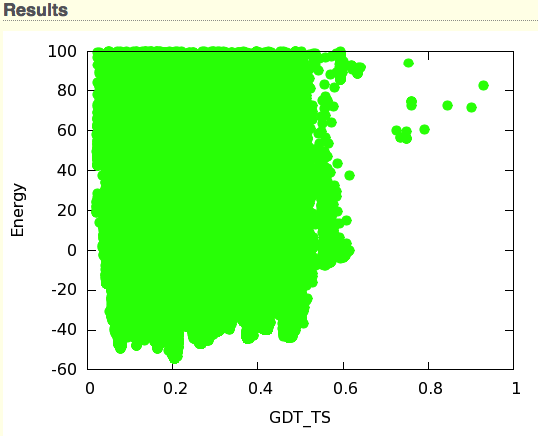
# Related work

On the Foldit website, the Foldit result has a section that shows the GDT\_TS vs Energy and RMSD vs Energy plots. These plots compare all the PDBs (Protein data bank files) to the native.

## 3.1 GDT\_TS vs Energy plot

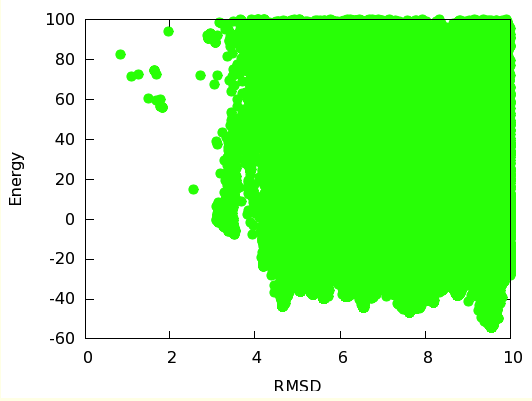
Below is an example of the GDT\_TS and RMSD plots for a puzzle: -

**1122: Revisiting Puzzle 85: Cell Adhesion Protein**



**Figure 15: GDT\_TS plot**

The above plot is the GDT\_TS vs Energy plot.[[18]](#endnote-18) The dots closest to 1 are closest to the native and the dots lower on the y-axis have the best energies. In the above plot, some players have tried to find protein structures with good energies but not so close to the native. For example, the dots above 0.2 have good energies but they are far away from the native. There are other dots closer to 1(native) but do not have as good energies as the ones above 0.2. Looking at the plot we can see that most players found proteins far away from the native and with good energies but we do not know the team they belong to. All the proteins are compared to the native and not to each other so there is no clustering. Clustering helps to see which proteins are similar. We also cannot see the protein structures and the name of the proteins.



**Figure 16: RMSD plot**

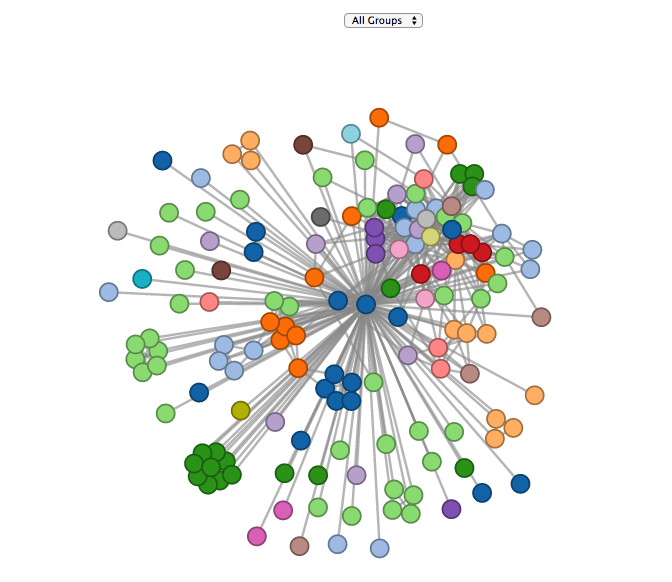
The above plot is the RMSD vs Energy plot. [[19]](#endnote-19) In case of RMSD, the dots closer to 0 are closest to the native and the dots lower on the y-axis have the best energies. In the above plot, the RMSD plot compares all the PDBs to the native. As we can see, the players tried to look for good energy in the same direction and did not think of diversifying. The peaks that we see are the players who thought they had the right protein structure and simply tried to improve the energy of the protein instead of trying to diversify along with improving the energy. If they had tried to diversify, their solutions could have been closer to the native.

If the players can pinpoint their location on the graph they would understand how they performed. But in GDT\_TS and RMSD, we cannot pinpoint a team. We just get an overall picture. All the PDBs are compared to the native and not to each other. So we do not know how similar/different the protein structures are from each other. We need a graph that would fulfill our above purposes.

But the GDT\_TS vs Energy graph was still not enough for comparison, so we decided to come up with an interactive visualization that would help us see the name of the PDB, its energy, the team it belongs to, and its approximate distance from the native and to each other. Clustering would help us see similar structures belonging to the same or different teams. The visualization would also help the Foldit developers see how the teams are performing and they could devise strategies to help the players improve in the upcoming puzzles.

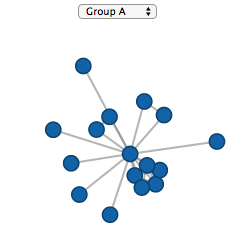
## 3.2 Previous analysis

Below are some screenshots of the first visualization that I came up with. The visualization was developed entirely in D3.



**Figure 17(A): All group selection for previous visualization (Force Directed Graph)**

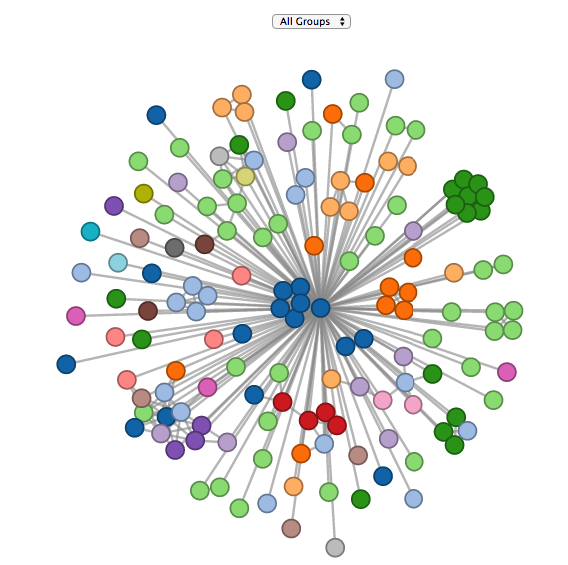
Above is a dataset of a Foldit puzzle that compares all the PDBs to the native and also clusters the similar PDBs. The circle in the center is the native. Similar proteins are seen clustered together. The clustering was based on a TMScore cutoff. For example, proteins with a TMScore value of 0.1 or less are said to be similar and can be clubbed together.

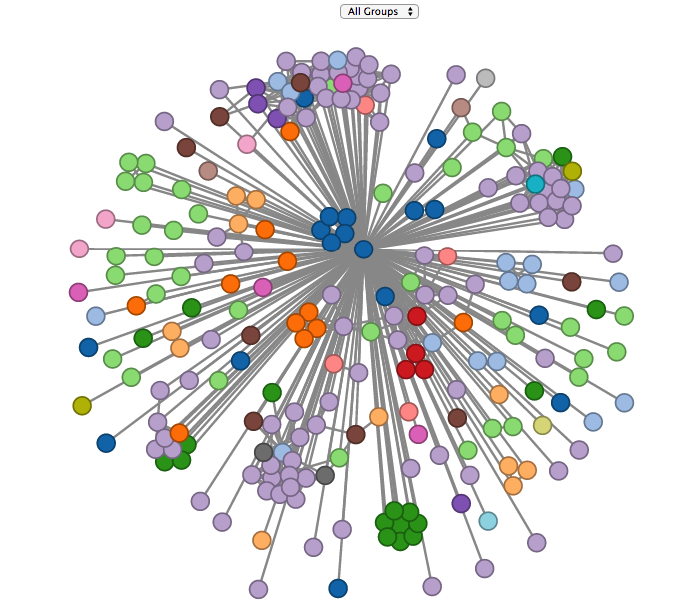


**Figure 17(B): Single group selection for previous visualization (Force Directed Graph)**

We can also select the PDB comparisons within a team(Group in this case). In the above figure, we select Group A from the drop down menu and only PDBs from team A are compared to the native.

Similarly, we ran tests on different datasets of Foldit puzzles shown below,





**Figure 17(C): Different datasets for previous visualization.**

## 3.3 Why it did not work (Drawbacks)

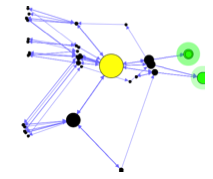
This visualization can compare PDBs to the native or the start but not both. All the PDBs could not be compared to each other, as it would mess up the visualization. Instead cutoffs had to be introduced to keep the visualization from not getting messy.

The visualization gets messy if the PDBs increase in number and inferring something gets difficult. The visualization would become messy especially because of the clustering. Consider a scenario in which there are more than 1000 PDBs. In that case, the PDBs would look clustered even though they are not. So the visualization does not accomplish everything we want, as it would only work for small number of PDBs.

## 3.4 Feature-Based Projections for Effective Playtrace Analysis

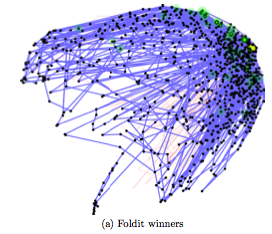
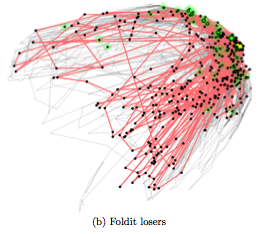
This was a very interesting tool developed by Yun-En Liu, Erik Andersen, Richard Snider, Seth Cooper, and Zoran Popovic at the University of Washington.[[20]](#endnote-20) This tool studied how the Foldit players played the game. The trajectory of how the Foldit players folded the protein from the start state to the end state was studied. By looking at the trajectory of the puzzle, one could see how the trajectory of the winners and losers looked like. MDS (Multi-Dimensional Scaling) was used to show how different the protein structures were from each other in the trajectory. The player’s problem solving strategies could be identified from this visualization. The trajectory showed where players did something different and ultimately reached the final state. The trajectory also showed how players strayed away from the final state and looked somewhere else for the solution and then ultimately quit.

The playtracer had drawbacks such as it could not handle large or continuous states. If the number of states increase considerably then it is difficult to understand what the players are thinking.



**Figure 18: Output of playtracer on a puzzle game.**

Above is an output of playtracer on a puzzle game. The yellow is the start state and green are the goal states. The arrows between states help in showing us the path from the start state to goal state. The size of the state tells us how many times a player visited that state. The states far away from the goal show us the incorrect moves made by players in solving the level.

**Figure 19: Playtracer trajectory for Foldit winners and losers.**

The above figures show the trajectories of the Foldit winners and losers. Looking at the above trajectories we can see than the goal state was near the start state but the players explored lot of areas till they reached the goal state that suggests the players did not immediately find the solution but had to explore a lot. The players who explored a lot were somewhere near the goal state at some point but later strayed away.

# Methods & Results

## 4.1 Python script

Since D3 is data driven, it needs a data file that has the data for the visualization. Similarly, MDS visualization needs a data file that has the information about every PDB that we see in the MDS plot. The data file has information such as energy, PDB file name, its group color, coordinates for every PDB dot to be plotted on the MDS graph, group name and player name. This information needs to be generated programmatically.

Some information needs to be extracted from the PDB files and some information like coordinates need to be generated firstly by doing pairwise comparison of PDBs using TMAlign. Once the two-dimensional TMScore matrix is generated, its values are inverted. Suppose in the visualization we need to show that the protein structures are totally different. This means visually they have to be apart. But their TMScore value is 0 as they are totally different. So we need to invert the value from 0 to 1 to show the proteins apart. This inverted TMScore matrix is passed to a MDS jar file that generates the coordinates to be plotted on the MDS graph. The energy, team name and player name information are extracted from the PDBs. Lastly, the script generates the data file with all the information needed by the MDS visualization. The data file is then passed on to the MDS visualization.

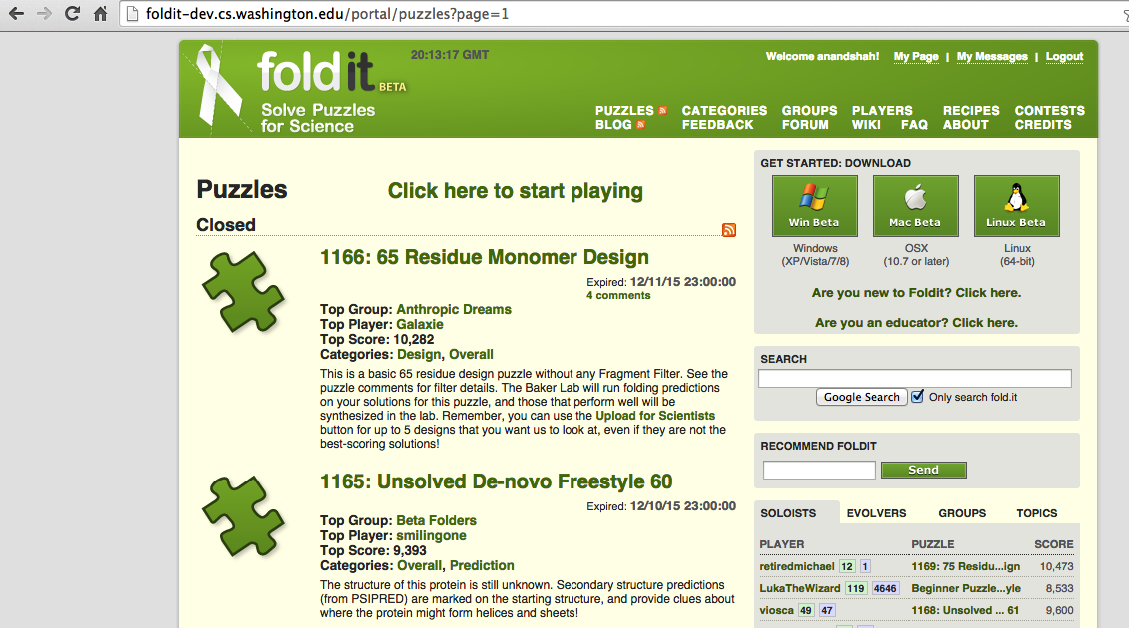
The purpose of writing the code in python was to make the backend code linux compatible, which was important since the Foldit backend code was also written in python and it runs in a linux environment. The python libraries made it easier for me to make the code less complicated. Once the python script was written it was merged with the existing Foldit code. Earlier, the Foldit code generated the GDT\_TS and RMSD plots in the results section of every puzzle. After merging the MDS python script with the existing Foldit code, the Foldit code now generates MDS plots along with the GDT\_TS and RMSD plots in the results section of every puzzle.

## 

## 4.2 MDS on the Foldit website

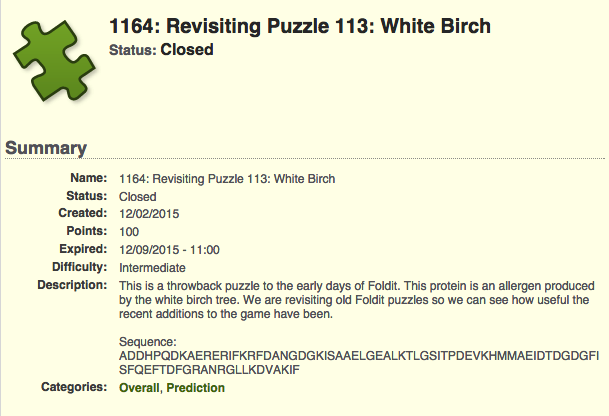
This is the link to the Foldit website

[http://Foldit-dev.cs.washington.edu/portal/puzzles?page=1](http://foldit-dev.cs.washington.edu/portal/puzzles?page=1)

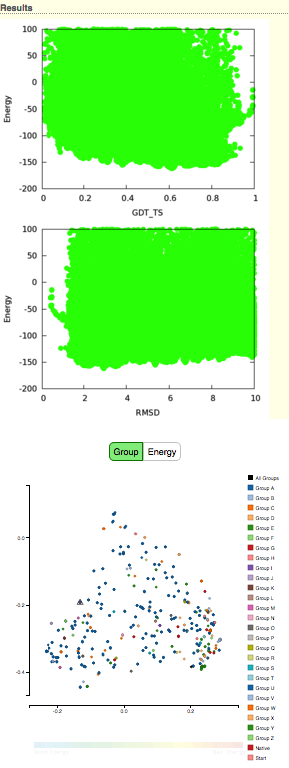
****

**Figure 20: Puzzle section on the the Foldit website**

We have a Puzzles section in the Foldit website. The puzzles section shows us all the puzzles that the Foldit players are currently working on and also the puzzles that are already closed. In the above figure we can see that puzzle 1166:65 is closed.

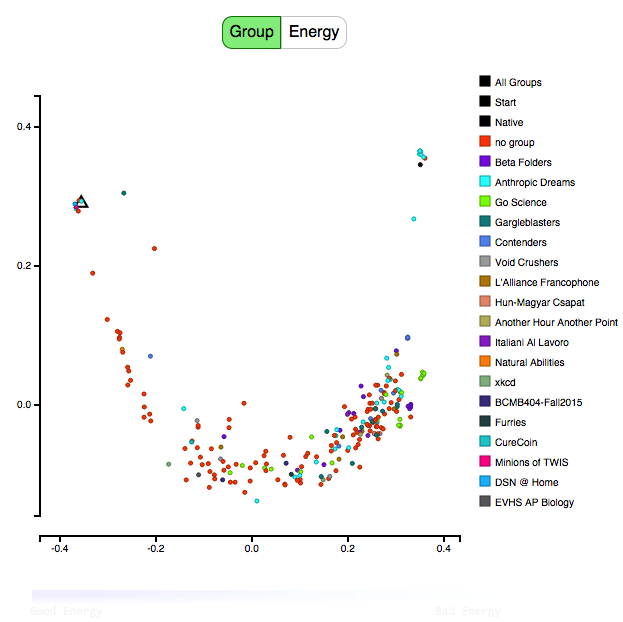


**Figure 21: A closed puzzle on the Foldit website**



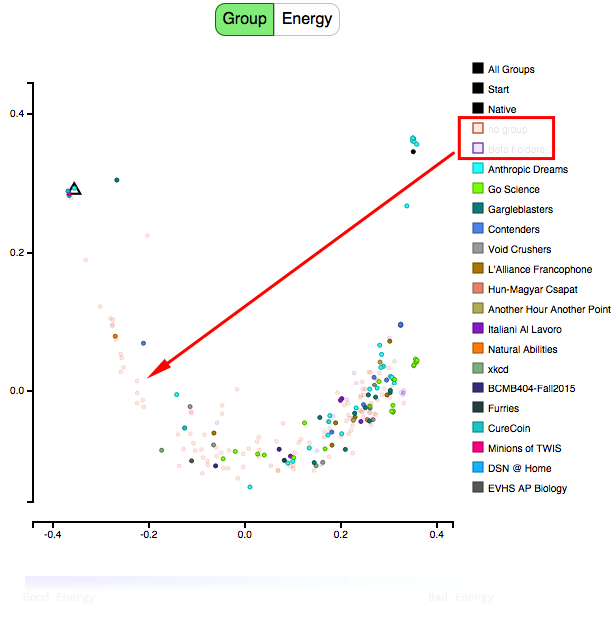
**Figure 22: GDT\_TS, RMSD & MDS plots for a puzzle.**

Above figure shows a closed puzzle 1164. We can see GDT\_TS, RMSD and MDS plots for that puzzle. Similarly, we can see GDT\_TS, RMSD and MDS plots for all the puzzles listed in the website. To see the GDT\_TS, RMSD and MDS plots in the puzzles one needs to have administrator privileges. Earlier there were only GDT\_TS and RMSD plots in the result section. Now we also have the MDS plots in the result section for every puzzle. The additional plot would help us analyze the puzzles better as the MDS plots overcome all the drawbacks of the GDT\_TS and RMSD plots.



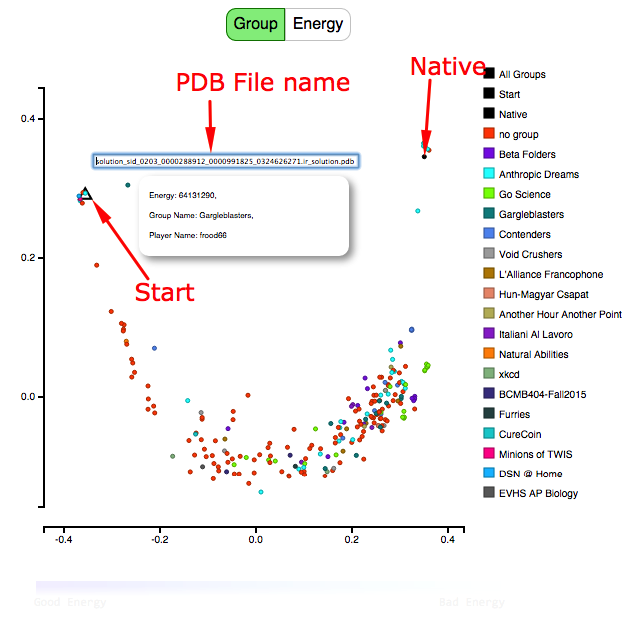
**Figure 23: MDS plot for a Foldit puzzle.**

Observe the MDS plot for a Foldit puzzle above. The legend on the right hand side shows the teams that participated in the puzzle. The dots seen above are PDBs that are colored according to the team they belong to. The coloring can be seen in the legend. The start structure is denoted by a triangle and the native structure is denoted by a black dot. Each PDB dot is the top scoring solution of a single player in the team. Looking at the plot we can see how the teams performed. We can see which team had their protein structures close to the native and which teams were far away from the native. The axes are the x & y coordinates for every PDB dot. We can find the coordinates for every PDB dot and then using the distance formula, the value that we get is approximately close to the inverted TMScore value between the PDBs. The Group tab is clicked in the above figure, which implies that the coloring of the PDB dots was done according to the group. The team names in the legend on the right hand side are sorted according to the top 10 performing teams in Foldit followed by rest of the teams in arbitrary order.



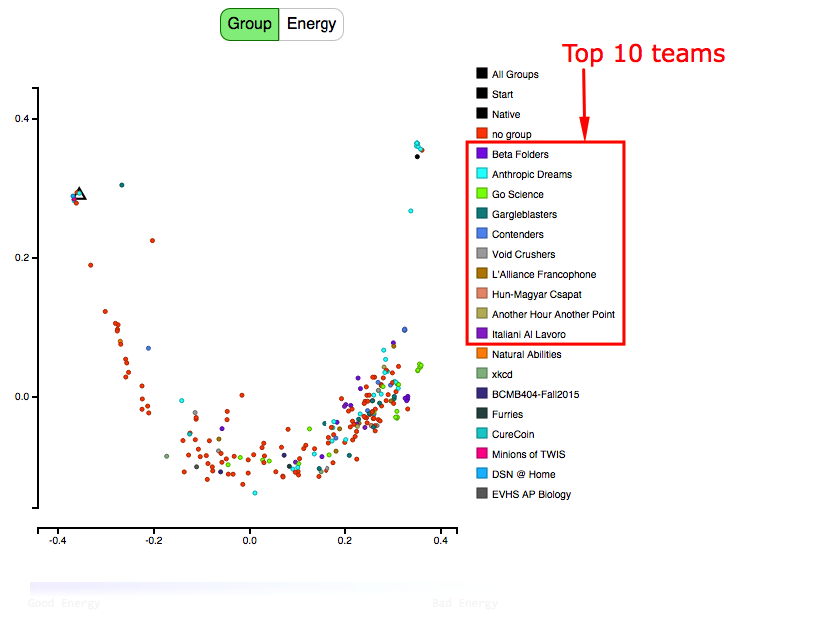
**Figure 24: Selecting-deselecting of teams in MDS**

We can select and unselect teams as we want by clicking on the team name in the legend. When we click on the team names, the PDB dots corresponding to the team names fade a bit (shown by the arrow pointing at the faded PDB dots). This helps us to focus on the teams that we want. Suppose we just want to see how the top 10 teams did in the puzzle, then in that case we can simply fade out the other teams. The “All Groups” in the legend fade or show all the teams at once when we click on it.



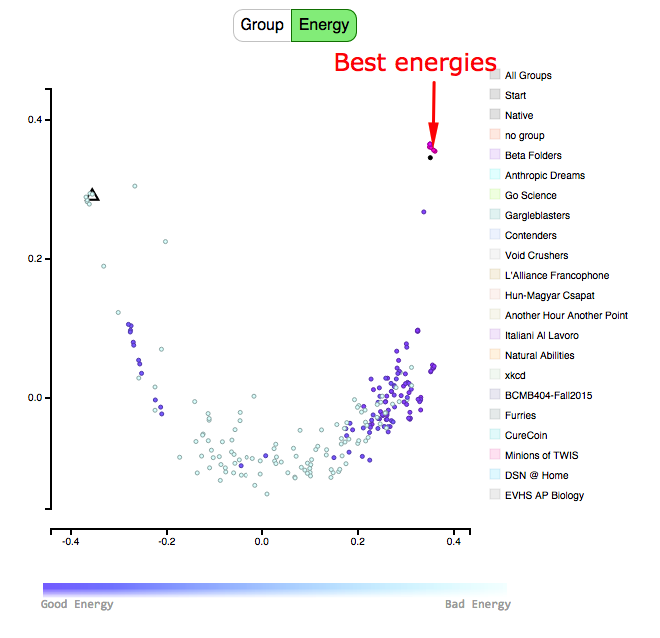
**Figure 25: Start, Native and PDB file name on the MDS plot**

When we hover over any PDB dot, we can see its Energy, the group (Team) it belongs to and the name of the player to whom the PDB belongs. If we click on the PDB dot we can see the name of the PDB solution file. An arrow above shows the Native and the Start PDBs.



**Figure 26: Top 10 teams in the MDS plot**

The figure above shows the top 10 teams. The “no group” above the top 10 teams are the PDBs that do not belong to any group. Hence they are placed above the top 10 teams.



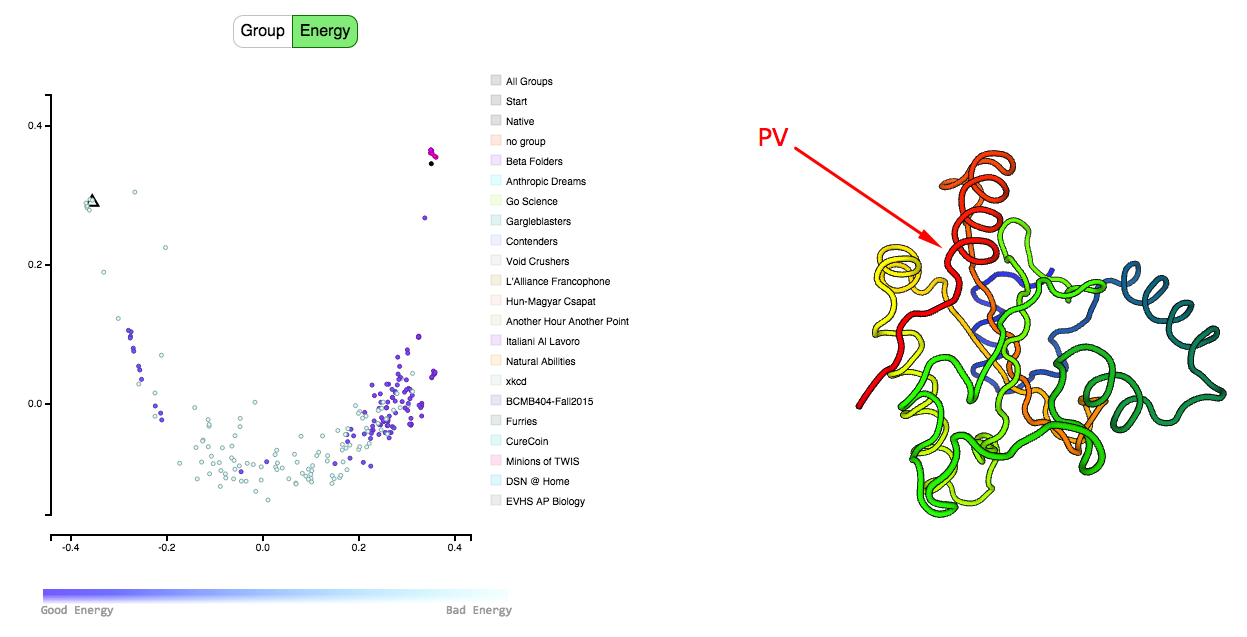
**Figure 27: PDBs with the best energies on the MDS plot.**

The above figure shows an energy scale that goes from good to bad. The PDB dots are colored according to the energy scale. For a protein structure, a negative energy is a good energy and a positive energy is a bad energy. Thus the more the negative energy, the better is the protein structure.

The PDB dots whose energies are above 0 are all colored faint blue. And all the PDB dots whose energy is below 0 are colored according to the energy scale. This is because we do not care about the protein structures whose energies are above 0. We only care about the PDBs whose energies are below 0.

When we click on the energy button, the legend boxes on the right hand side of the plot fades out and the energy scale shows up. Same thing happens when we click on the Group button, that is, the energy scale fades out and the legend boxes on the right hand side of the plot shows up.

In the above figure, the PDB dots next to the native are colored dark purple (as shown above), which means they have the best energies. The PDB dots that are colored light blue are those PDBs who have energies above 0. The selecting-unselecting of teams still work if we click on the team names in the legend. And we can still see the PDB information like the Energy, Group (team) name and Player name when we hover over a PDB and we can also see the name of the PDB file if we click on the PDB dot.



**Figure 28: PV protein viewer in MDS.**

Above is the MDS site hosted on the Umassd server. This is the second version of MDS with the protein structure visualizer. The first version which is on the Foldit website as of now does not have the protein structure visualizer.

For the protein structure visualizer to show the proteins, the PDBs have to be physically present on the Foldit server. The PDB files cannot exist on the Foldit server as the Foldit server does not have that much amount of space to store all the PDBs. The PDBs could also be fetched from another location on the click of a PDB dot. But this would require large amount of bandwidth as there will be lot of clicks and PDBs will be fetched at runtime. Thus, for now we will not be able to show the PDBs on the Foldit server. So temporarily we have hosted MDS with the protein structure visualizer on the Umassd server where we have stored the PDB files.

The Foldit team is in the process of hosting the Foldit website on Amazon AWS. Once Foldit is on AWS, we will be able to store the PDBs there and the protein structure visualizer in MDS will be able to fetch the PDBs directly from the server.

The purpose of MDS is to find a pattern in which the Foldit players play the game and which teams usually perform the best. Sometimes a protein structure away from the native could have a very good energy. We can look at the protein structure in PV and see why the protein structure has good energy even though it is away from the native.

## 4.3 PV in MDS

In the MDS plots, we can see how similar/different the protein structures are from each other. But we would not know how they are different unless we see the proteins. It is quite possible that there are PDBs away from the native yet have good energies. In this case, we would like to see why the protein structures are so different from each other yet have good energies. So we need a protein structure visualizer that would show us the protein structure on the click of the dot in the MDS plot.[[21]](#endnote-21)

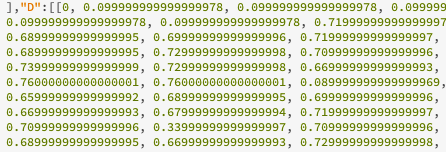
So I plugged in a tool called PV which is a JavaScript tool that shows a three dimensional view of a protein structure. PV is an open source JavaScript tool. PV is a super-fast tool unlike other JavaScript tools like JLMoL which takes time to load and has comparatively slower interaction. PV is also easy to integrate inside any website and does not require any plug-ins to be installed. PV uses WebGL for rendering. The code for PV is lightweight. The complete minified code is around 100kb. Below is the screenshot of PV integrated in the MDS website.

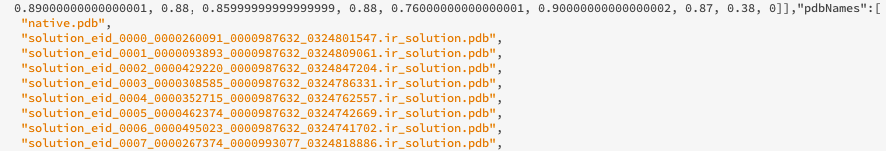
PV is unlike other java based protein viewers. Java based protein viewers are difficult to integrate inside a website. Java based PVs need lot of updates before we can use the applet which is kind of frustrating. We need a tool which we can easily plugin and which does not require any updates of any sort. Opening the applets also sometimes freezes the browser that shouldn’t happen at any cost. PV has no such issues. Also PV has much better graphics than JSMol and any other protein viewers.

## 4.4 Data file for MDS visualization

Since D3 is data driven, it needs a JavaScript file as a data file to load all the data related to the visualization. Below is the format in which the data is stored.

****





**Figure 29: Data for a PDB dot in the MDS data file.**

The “X” in the above figure holds the group name (team name), player name, energy of the PDB, color of the PDB and the X & Y coordinates of the PDB dot. The “D” holds the two-dimensional inverted TMScore matrix. The “PDBNames” holds the names of the PDBs.

# 5. Conclusion & Future work

The MDS plots generated for every puzzle can be used to study the pattern in which the players fold the protein. We can also see which group did the best job at finding the right protein structure.

The future work would involve finding a pattern in which the groups solve the puzzle and also identify the group doing a better job than the others. We can also study the protein structures of players that might be better than the native. MDS can also be used to compare Rosetta solutions. We could find strategies by which Foldit players can improve their solutions. We can encourage the players to diversify their solutions. If the Foldit players do better than Rosetta, then that would show a loophole in Rosetta and we can make Rosetta better by improving the Rosetta algorithm.

# References

1. ­­[] <http://cavanagh-lab.bch.ncsu.edu/bch701/Protein_folding_class.pdf> [↑](#endnote-ref-1)
2. [] **Algorithm discovery by protein folding game players**

   Firas Khatib , Seth Cooper , Michael D. Tykaa ,,Kefan Xub , Ilya Makedonb , Zoran Popovićb , David Bakera,c,1, and Foldit Players [↑](#endnote-ref-2)
3. [] **Are predicted structures good enough to preserve functional sites?**

   Liping Wei1, Enoch S Huang2 and Russ B Altman1\* [↑](#endnote-ref-3)
4. [] **Data-mining Protein Structure Databanks for Crystallization patterns of Proteins.**

   Homayoun Valafar\*, James H. Prestegard\*\*, Faramarz Valafar\*\*\* \* Southeast collaboratory for Structural [↑](#endnote-ref-4)
5. [] **MULTIPLE STRUCTURE ALIGNMENT BY OPTIMAL RMSD IMPLIES THAT THE AVERAGE STRUCTURE IS A CONSENSUS**

   Xueyi Wang\* and Jack Snoeyink Department of Computer Science, University of North Carolina at Chapel Hill [↑](#endnote-ref-5)
6. [] **STRUCTURE COMPARISON AND ALIGNMENT**

   Marc A. Marti-Renom, Emidio Capriotti, Ilya N. Shindyalov, and Philip E. Bourne [↑](#endnote-ref-6)
7. [] **PDB-by-RMSD: The New Service and Tool for Searching Protein Structures**

   A. S. Shevnin, A. D. Semenov, Yu. B. Porozov [↑](#endnote-ref-7)
8. [] **fRMSDPred: Predicting local rmsd between structural fragments using sequence information** [↑](#endnote-ref-8)
9. [] **Pcons5: combining consensus, structural evaluation and fold recognition scores**

   Bjo¨ rn Wallner and Arne Elofsson [↑](#endnote-ref-9)
10. [] **Applying Undertaker Cost Functions to Model Quality Assessment**

    John Archie and Kevin Karplus September 5, 2008 [↑](#endnote-ref-10)
11. [] **A polynomial time algorithm for computing the area under a GDT curve**

    Aleksandar Poleksic\* [↑](#endnote-ref-11)
12. # [] TM-align: a protein structure alignment algorithm based on the TM-score

    [Yang Zhang](http://www.ncbi.nlm.nih.gov/pubmed/?term=Zhang%20Y%5Bauth%5D) and [Jeffrey Skolnick](http://www.ncbi.nlm.nih.gov/pubmed/?term=Skolnick%20J%5Bauth%5D)\* [↑](#endnote-ref-12)
13. # [] Fr-TM-align: a new protein structural alignment method based on fragment

    # alignments and the TM-score.

    [Shashi Bhushan Pandit](http://www.ncbi.nlm.nih.gov/pubmed/?term=Pandit%20SB%5Bauth%5D)1 and [Jeffrey Skolnick](http://www.ncbi.nlm.nih.gov/pubmed/?term=Skolnick%20J%5Bauth%5D) [↑](#endnote-ref-13)
14. # [] Fr-TM-align: a new protein structural alignment method based on fragment alignments

    # and the TM-score.

    Shashi Bhushan Pandit and Jeffrey Skolnick [↑](#endnote-ref-14)
15. [] **Kruskal, J.B. and M. Wish 1978. *Multidimensional Scaling*. Sage.**

    * Moore, Rod 1990. "Ethnographic assessment of pain coping perceptions." *Psychosomatic Medicine 52:171-181.*
    * Blank & Mattes "Sugar and Spice: Similarities and Sensory Attributes" *Nursing Research 39(5):290-293*
    * Wexler & Romney. 1972. "Individual Variations in cognitive structures." in Romney, Shepard & Nerlove eds. *Multidimensional Scaling: Theory and Applications in the behavioral sciences, Vol II.* Seminar Press.

    <http://analytictech.com/networks/mds.htm> [↑](#endnote-ref-15)
16. [] **MULTIDIMENSIONAL SCALING**

    *Forrest W. Young, University of North Carolina* [↑](#endnote-ref-16)
17. [] **D 3 : Data-Driven Documents**

    Michael Bostock, Vadim Ogievetsky and Jeffrey Heer [↑](#endnote-ref-17)
18. # [] Modeling Disordered Regions in Proteins Using Rosetta

    Ray Yu-Ruei Wang, Yan Han, Kristina Krassovsky, William Sheffler, Michael Tyka, David Baker . [↑](#endnote-ref-18)
19. [] **Evaluating conformational free energies: The colony energy and its application to the problem**

    **of loop prediction.**

    Zhexin Xiang, Cinque S. Soto, and Barry Honig [↑](#endnote-ref-19)
20. [] **Feature-Based Projections for Effective Playtrace Analysis**

    Yun-En Liu, Erik Andersen, Richard Snider, Seth Cooper, and Zoran Popovic´

    Center for Game Science Department of Computer Science & Engineering, University of Washington {yunliu, eland, sniderrw, scooper, [zoran}@cs.washington.edu](mailto:zoran%7d@cs.washington.edu) [↑](#endnote-ref-20)
21. [] <https://biasmv.github.io/pv/>

    <https://zenodo.org/record/20980#.Vn86-vkrJaQ> [↑](#endnote-ref-21)