ProMuteHT : A High Throughput Compute Pipeline for Generating Protein Mutants *in silico*

Erik Andersson
Western Washington University
516 High Street
Bellingham, WA 98225
ander625@wwu.edu

Filip Jagodzinski
Western Washington University
516 High Street
Bellingham, WA 98225
filip.jagodzinski@wwu.edu

ABSTRACT

Understanding how an amino acid substitution affects a protein's structure is fundamental to advancing drug design and protein docking studies. Mutagenesis experiments on physical proteins provide a precise assessment of the effects of mutations, but they are time and cost prohibitive. Computational approaches for performing *in silico* amino acid substitutions are available, but they are not suited for generating large numbers of protein variants needed for high-throughput screening studies. We present **ProMuteHT**, a program for high throughput *in silico* generating user-specified sets of mutant protein structures with single or multiple amino acid substitutions. We combine our custom mutation algorithm with side chain homology modeling external libraries, and generate energetically feasible mutant structures. Our efficient command-line invocation syntax requires only a few arguments to specify large datasets of mutant structures. We achieve quick run-times due to our hybrid approach in which we limit the use of costly energy calculations when mutating from a large to a small amino acid. We compare our mutant structures with those generated by FoldX, and report faster run-times. We show that the mutants generated by **ProMuteHT** are of high quality, as determined via all-atom RMSD measurements for existing mutant structures in the PDB.

CCS CONCEPTS

•**Applied computing** → **Molecular structural biology;** *Bioinformatics;*

KEYWORDS

Mutagenesis; in silico; Modeling; Protein Structure

1 INTRODUCTION

Knowing to what extent a mutation affects a protein's stability can aid in drug design studies that aim to deliver pharmaceutical solutions for combating diseases caused by protein mutants [24]. Protein-compound interactions play a vital role in drug design studies, and generating *in silico* mutants is a major component driving such work [20].

One approach for studying the effects of a mutation is to perform *in vitro* experiments by engineering amino acid substitution directly in a physical protein, and then solving the structure via X-ray crystallography. However, that is impractical because some variants are difficult to crystallize. Moreover if a large set of mutant variants must be assessed, working with physical proteins is prohibitive because a single experiment to infer the role of just one mutation can be costly and may take weeks, or more, of wet-lab work. Alternatively, it is possible to infer the effect of a mutation in a physical protein without the need to resolve the structure. Free energy of unfolding experiments that use denaturants involve measuring the extent to which the wild type (WT), non-mutated, protein denatures relative to the mutant, and a quantitative assessment of the effects of the mutation(s) is attained via the Schellman equation [25]. However that approach, too, involves wet-lab work, and assessing the effect of more than a small handful of mutations is impractical.

To help complement and inform mutation studies performed on physical proteins, computational methods are available. Some methods reason about the effects due to changes to side-chain conformations [9, 15, 23], others rely on heuristic energy functions or database-derived potentials [10, 18], while others still leverage insights from large data sets of homologous proteins [5, 27, 28]. Machine learning (ML) approaches developed for this need exhibit great variety as well, with some relying on Vector Machine methods [7, 14], while others utilize Random Forest and similar approaches [6, 13, 16, 19].

Most of these computational methods do not output structure files of protein mutants, in spite of a need by protein docking and drug design studies for many structures for high throughput screening purposes. Moreover, the majority of prediction and modeling methods are most suited for generating and assessing structures with single amino acid substitutions. It is because of these limitations of existing tools that we have developed a stand-alone program that efficiently generates many protein mutants *in silico.*

2 RELATED WORK

Generating and modeling mutations in protein structures has been done a variety of ways over the years. Some tools require extensive human intervention, while others include near-automated capabilities. In this section we survey a handful of tools for modeling and generating amino acid substitutions.

ACM-BCB'17, August 20-23, 2017, Boston, MA, USA.

978-1-4503-4722-8/17/08…$15.00
DOI: https://doi.org/10.1145/3107411.3116251

In early work, amino acid substitutions were simulated not by altering protein structure files, but by modifying energy and structure parameters of simulations of proteins. Motivated by a need to predict how mutations influence molecular functions, You, *et. al* [29], indirectly studied the effects of mutations on bacteriophage T7 molecules by altering molecule parameters in their simulations. In such studies, no structure files were ever generated.

More recent tools for generating mutant structures with amino acid substitutions are available. PyMol [8] offers a rich set of Graphical User Interface (GUI) tools for generating amino acid substitutions. A user manually modifies individual residues with the aid of a rotomer library and a visual guide showing possible steric clashes due to the choice of the side chain selected. This approach is amenable to creating only a small set of mutants, as each one has to be created by-hand. Another tool, Swiss PDB viewer [12], is unfortunately not available in Linux, and thus cannot be integrated into a custom compute pipeline for generating large sets of mutants.

A few tools for generating mutants afford components of automation, but they still require a more-than-desired amount of human intervention. Some of these are dependent on costly energy calculations which reduces their applicability when thousands of mutants are needed. One such tool is the `buildmodel` command in the FoldX software suite [26]. The FoldX server and applications were developed around an empirical force field for evaluating the effects of mutations. The main component of the software calculates free energy values for a macromolecule using a linear combination of 9 empirical energy terms, including hydrogen bond energies, energies due to steric clashes, and energies arising from van der Waal forces. The `buildmodel` option in FoldX is a relatively new feature of the suite of tools, and requires composing a list by-hand of all mutations that are needed. Most importantly, when multiple mutations in a single structure are *in silico* generated, the energy terms for the force field are calculated for each substitution, which creates a computational bottleneck when many *in silico* amino acid substitutions in a single structure are sought.

3 MOTIVATION

Thus, existing tools are available for *in silico* creating protein structures with amino acid substitutions. In the case of those tools that require a user to manually interact with a GUI, they are not practical for creating large sets of mutant structures. Alternatively, there exist tools for generating good mutant structures, but they nonetheless have limitations. For example FoldX requires creating a mutation list file by hand. In addition, most tools rely on energy and force field calculations, which prevents them from being used to generate massive sets of mutants with multiple mutations each, despite a need for such structures across different research efforts. For example, upwards of 10^5 simulation steps are commonly used to generate and score decoys for studying protein-protein docking interactions [11], while machine learning approaches for studying protein-ligand interactions are dependent on vast counts of complexes for screening drug targets [1].

In this work we present a new, hybrid compute pipeline for generating *in silico* mutations, whose reliance on costly energy calculations we reduce as much as possible. We are motivated by a need to generate large sets of mutant structures. To that effect, we have developed **ProMuteHT**, with two main goals in mind :

(1) Ease of specifying which mutants should be generated
(2) Reducing the need on energy calculations to improve performance when scaling to generate upwards of hundreds of thousands of mutant structures.

4 METHODS – COMPUTE PIPELINE

There are several components to our mutant generation process, summarized in Figure 1. Upon input of user-supplied parameters informing what *in silico* mutations are desired, PDB and FASTA file are fetch and preprocessed. The mutant generation step includes invoking one of two modules for each mutation that is specified, depending on whether the mutation is from a large to a small amino acid, or small to large amino acid. All mutant structure files are energy minimized to account for unfavorable energetic terms that might have been introduced during the *in silico* mutation.

4.1 Parameter Input, and Preprocessing

There are several command-line input formats accepted by **proMuteHT**. One permits a user to specify a comma-separated list of residue numbers, and/or range of amino acids, as well as the specific amino acid substitution(s) to be performed. The general syntax is of the following form:

$$\texttt{PDBID chain}_s\texttt{:chain}_e\ \texttt{res}_s\texttt{:res}_e\texttt{,res}_n\{^*X\}\ \texttt{subs}$$

where `PDBID` is the 4 alphanumeric structure ID from the Protein Data Bank [3], $\texttt{chain}_s$ and $\texttt{chain}_e$ are used to specify the chain(s) in the protein on which mutations should be generated, $\texttt{res}_s$ and $\texttt{res}_e$ are used to specify a range of residues to be mutated, and $\texttt{res}_n$ are individual residues delimited by comma(s). Providing a single letter amino acid abbreviation in the residue designation portion of the command permits specifying combinatorial *n*-wise sets. The use of the `:` permits specifying ranges (inclusive), while the optional

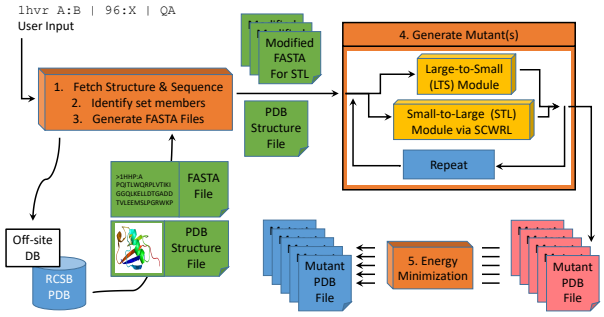

Figure 1: ProMuteHT fetches a protein structure and its FASTA file from the PDB (1), enumerates a mutant list per the user's input parameters (2), processes the FASTA file as needed by SCWRL [17] when a small-to-large mutation is required (3), and iteratively generates mutant files (4) which then are energy minimized (5).

parameters `X` and * designate *all*, and *hold fixed*. The `subs` field indicates which amino acid substitutions should be performed, what we call the *targets*, and consists of the single letter abbreviations for the 20 naturally occurring amino acids, as well as the lower case characters `pol`, `char`, and `phobic` to specify amino acids that are polar (Gln, Asn, His, Ser, Thr, Tyr, Cys, Met, Trp), charged (Arg, Lys, Asp, and Glu), and hydrophobic (Ala, Ile, Leu, Phe, Val, Pro, and Gly). Consider the below two invocations:

```
1hvr A:B 96:X A                (1)
2ped A:A 3,10:16 phobic        (2)
```

Invocation (1) specifies mutate chains A through B of PDB structure 1hvr, residues 96 through the end (residue 99), to residues Alanine, and would result in a total of $2_c \times 4_r \times 1_t = 8$ mutants, where the `c`, `r`, `t` subscripts refer to the number of chains, residues, and targets. Invocation (2) specifies mutate chain A of PDB structure 2ped, residues 3, and 10 through 16, and mutate each of them systematically to the hydrophobic residues, for a total of $1_c \times 8_r \times 7_t = 56$ mutants. More intricate examples are give in Section 5.1.

When the input parameters are parsed, checks are performed for the values provided by the user. A warning message is output and the program terminates whenever any of the following occurs:

- The PDB ID supplied does not adhere to the 4 alphanumeric character format in use by the Protein Data Bank
- There is no entry in the PDB for the provided ID
- The specified chain (or a chain in a range specified), and/or residues do not exist in the PDB structure file
- There is ambiguity in the formatting of the syntax specifying which mutants to generate.

If the input arguments would result in more than 250 mutants being generated, the user is prompted to confirm continue.

4.2 Generating Mutants

ProMuteHT is composed of several parts, including custom scripts and our in-house developed software, integrated among off-the-shelf open access tools and libraries. In an effort to reduce as much as possible the need for force field calculations, we have identified the following two broad types of amino acid substitutions :

(1) large-to-small (LTS) amino acid substitutions
(2) small-to-large (STL) amino acid mutations.

Our iterative program for generating mutant PDB structure files is made up of LTS and STL modules for performing these tasks. In the case that multiple mutations in a single structure are desired, they are performed one-after-the-other. The output structure that results from invoking either the LTS or STL module is the input structure for the next amino acid substitution in the list of mutations being performed *in silico* (*Repeat* step in Figure 1).

The LTS module removes atoms from a PDB file to simulate large-to-small amino acid substitutions. No energy minimization of any kind is involved. See Figure 2 for a schematic. As an example, mutating Valine or Leucine to Alanine involves removing the CG1 and CG2 atoms from Valine, and CG, CD1 and CD2 atoms from the Leucine structure file. Mutating Valine or Leucine to Glycine involves removing all of the side chain atoms, including the CB

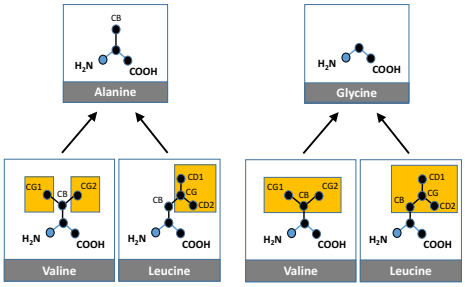

Figure 2: Large-to-small (LTS) mutations in ProMuteHT do not require energetics calculations. The *in silico* mutation involves retaining only the atoms of the target, smaller, residue. To mutate Valine or Leucine to Alanine, we remove the carbon atoms at the γ and δ positions (yellow boxes).

atom. The CB, CG1, etc. are the Cβ, Cγ, etc. atom nomenclature terms in use in PDB files. Other LTS mutations, for example to Serine, follow a similar methodology. Although a void is created when this procedure is performed, especially when mutating a core residue from a large to a small one (Phenylalanine to Glycine, for example), the energy minimization step (Section 4.3) addresses this.

When a small residue is mutated to a larger one, the STL module is invoked. It relies on the freely available [17] software that makes predictions about a side chain's orientation. SCWRL uses rotamer libraries [4] based on kernel density estimates to make predictions about side chain confirmations. Among several other metrics, it is dependent on a hydrogen bonding function and a van der Waals interaction potential. **Although SCWRL is fast, we want to minimize its use nonetheless because even brief energy calculations can amount to lengthy delays when tens of thousands of mutations are being performed**. SCWRL's input is a PDB file, and the corresponding FASTA sequence with edits specifying desired side chain replacements. In Step 3 of the compute pipeline (Figure 1), a modified FASTA file is generated for any mutant desired which requires a small-to-large mutation. For example, if the FASTA sequence of the wild type is GAGGAFN, and the third residue's side chain is being mutated to aspartic acid, then the modified FASTA file would be edited to contain GAdGAFN (we underline the changes).

In the case of multiple-chain proteins, we do not use SCWRL as-is, because in those cases it often cannot make a side chain replacement. In that case, our software extracts each chain in the PDB file, and uses SCWRL to mutate the residue in the chain where the mutation is being performed. **ProMuteHT** then combines the un-mutated chain(s) and the altered one, and the end product is a multi-chain protein with the desired mutation.

4.3 Energy Minimization

Performing energy minimization is especially necessary when a small-to-large mutation is involved. That is because SCWRL does not take into account steric clashes that arise due to the introduction of new side chain atoms into the space occupied by the smaller

residue's fewer atoms. Although mutating a large residue to a small one using our LTS module does not introduce steric clashes, there is nonetheless a void introduced, so energy minimization is performed in those cases, too. We rely on the NAMD [22] software to perform 200 energy minimization steps, requiring a few seconds of computation time, depending on the protein's size. We chose 200 steps from empirical experiments in which we observed that most structures by that time had settled enough so that high velocity atoms were no longer present.

5 RESULTS

To demonstrate the utility of **ProMuteHT**, we tallied its run-times, assessed the quality of the mutants generated, and also compared our approach with the mutation engine in FoldX. We used a quad core Intel Xeon E5-2620 processors at 2.0GHz, with 16GB of RAM.

5.1 ProMuteHT v.s. FoldX, Speed

In Table 1 we show four invocations of **ProMuteHT** for generating data sets with 13, 171, 342, and 418 mutants. The *real* run-times were measured using the linux *time* command.

Table 1: Run Times for sample invocations of ProMuteHT.

Invocation	# res	# muts	run-time (s)
`1hhp A:A 3,7,10:20 G`	99	13	21.9
`1crn A:A 12:30 pol`	46	171	220
`1pef A:A X:X X`	18	342	407
`1hvr A:B 20:30 X`	198	418	586

To compare the run-times of **ProMuteHT** with FoldX, we invoked both programs to generate two sets of mutants from PDB files 1hhp (99 residues) and 1pef (18 residues). We created by hand the required *mutant-file.txt* parameter files for specifying in FoldX what mutations to generate. We found that to be prohibitive, because not only must the target residue be specified, but FoldX also requires that the parameter file specify the non-mutated residue at the substitution point. In comparison, the commands for invoking our software for generating the equivalent mutants are the following:

```
1hhp A:A 19G*,20:25 A        (3)
1pef A:A X:X G               (4)
```

Invocation (3) specifies mutate chain A of structure 1hhp, residues 20 through 25 to an Alanine, paired with residue 19 being mutated to a Glycine. This generated the following 6 mutants:

- Mutant_1 : res. 19 mutated to Glycine, and 20 to Alanine
- Mutant_2 : res. 19 mutated to Glycine, and 21 to Alanine
- Mutant_2 : res. 19 mutated to Glycine, and 22 to Alanine
- Mutant_3 : res. 19 mutated to Glycine, and 23 to Alanine
- Mutant_4 : res. 19 mutated to Glycine, and 24 to Alanine
- Mutant_5 : res. 19 mutated to Glycine, and 25 to Alanine

Invocation (4) specifies mutate all residues in Chain A of 1pef to a Glycine. The mutants for 1hhp and 1pef are shown in Table 2. The run-times reveal that **ProMuteHT** has far overall lower run-times than FoldX when generating the same mutants. We suspect that is because the `buildmode` of FoldX relies heavily on energy calculations, and that even in the case of a small protein (1pef), the computations required are non-trivial. However our *in silico* LTS mutation process does not rely on energy calculations except for a brief minimization run after all substitutions have been performed. Although the speedup is only 2-3X, such a speedup would become significant if thousands of mutants were being generated.

Table 2: FoldX vs ProMuteHT run times for *in silico* generating two sets of mutants with 5 and 18 structures each.

protein	# muts	mutants (comma delimited)	Run-Times (s) FoldX	Run-Times (s) ProMuteHT
1hhp	6	L19G and K20A, L19G and E21A, L19G and E22A, L19G and L23A, L19G and L24A, L19G and D25A	82.5	12.3
1pef	18	E1G, Q2G, L3G, L4G, K5G, A6G, L7G, E8G, F9G, L10G, L11G, K12G, E13G, L14G, L15G, E16G, K17G, L18G	66.5	32

5.2 ProMute v.s. FoldX, Quality

To validate the quality of the amino acid substitutions performed by **ProMuteHT**, we generated several mutants whose structures we then compared to actual mutant structures in the PDB. We also compared the quality of our structures to those generated by FoldX.

We searched the PDB and selected 20 wild type, mutant pairs, involving substitutions to all possible naturally occurring amino acids. Eleven of those entries include 2lzm as the wild type. That is because Matthews *et al.* have generated and resolved the structures of many variants of Lysozyme from Bacteriophage T4 [2, 21]. We used **ProMuteHT** to *in silico* mutate the wild type of each pair, performing the same mutation that exists in the mutant structure. For each generated mutant, actual mutant pair, we measured the all-atom Root Mean Squared Deviation (RMSD), for both the full structure and the mutated residues only.

The run-times and comparative results of *in silico* mutating single amino acids are shown in Table 3. The **RMSD all** column lists the all-atom RMSD measurements for the mutants generated by FoldX (FX) and **ProMuteHT** (PM) for the entire structure compared to the known PDB mutant structure. The **RMSD res** column lists the all-atom RMSD measurements for the mutants generated by FoldX (FX) and **ProMuteHT** (PM) for just the atoms of the residue that was mutated. Yellow, red, and green entries for the PM columns identifies mutant structures we generated that were equally as good, worse, or better than those generated by FoldX, when compared to the actual mutant structures available in PDB files. Notice that even in those cases where our generated structures were not as good as those generated by FoldX, the RMSD values are nonetheless similar. When the RMSD measurements were calculated using only the mutated residue's atoms, the quality of the mutants generated by **ProMuteHT** was overall higher than those generated

Table 3: The quality of variants output by ProMuteHT was assessed by comparing *in silico* generated mutants with mutant structures from the PDB with equivalent mutations. Res=residue in the polypeptide chain, AA=Amino Acid for the wild type and Mutant for the indicated Res, and ASA=Accessible Surface Area. RMSD=Root Mean Squared Deviation between the actual (from PDB) and *in silico* generated mutants. Colors specify structures output by ProMuteHT (PM) that were worse (red), the same (yellow), or of better (green) quality than those generated by FoldX (FX).

WT Structure				Mutant		RMSD all		RMSD res	
PDB	res	AA	ASA	AA	PDB	FX	PM	FX	PM
1lz1	56	I	0.3	A	2hec	0.21	0.21	0.04	0.04
2lzm	149	V	0.0	C	237l	0.17	0.19	0.05	0.04
1lz1	110	V	60.1	D	1gfu	0.22	0.26	0.91	1.10
2lzm	105	Q	32.5	E	1l98	0.22	0.25	0.21	0.14
1lz1	54	Y	8.1	F	1wqq	0.21	0.22	0.77	0.06
1lz1	59	I	1.6	G	2hee	0.22	0.22	0.03	0.03
2lzm	157	T	41.8	H	1l09	0.08	0.11	0.85	1.12
2lzm	99	L	0	I	1l92	0.13	0.15	0.26	0.07
2lzm	120	M	16.4	K	232l	0.19	0.22	0.21	0.17
2lzm	6	M	0	L	230l	0.20	0.22	0.68	0.35
2lzm	99	L	0	M	1l93	0.13	0.15	1.10	1.06
2lzm	2	V	0	N	1gfe	0.22	0.24	0.85	1.36
2lzm	3	I	10.7	P	1l96	0.21	0.24	0.21	0.17
2rn2	48	E	10.8	Q	1rdb	0.19	0.23	0.18	0.17
1lz1	2	V	0	R	1gfg	0.22	0.25	0.85	0.77
2lzm	42	A	0	S	206l	0.15	0.15	0.04	0.03
1lz1	56	I	0.3	T	1oua	0.22	0.41	0.08	0.15
2lzm	44	S	57.5	W	216l	0.90	0.96	0.01	0.10
1lz1	56	I	0.3	V	2bqd	0.23	0.26	0.11	0.10
1lz1	110	V	60.1	Y	1gft	0.27	0.31	0.03	1.31

by FoldX, indicated by the green entries. Also notice that some residues were deeply buried (low Accessible Surface Area), and when mutating them from small to large amino acids, the *in silico* generated structures nonetheless had good alignment with the actual mutant structures from the PDB. **Overall, the quality of the mutant structures generated by ProMuteHT is comparable – and often better – to the quality of the structures generated by FoldX, while our run times are better** (Section 5.1).

We also assessed the quality of mutants generated by **ProMuteHT** when multiple amino acid substitutions were performed *in silico*. As with our validation of mutants with single mutations, we compared the mutants we generated with PDB files containing corresponding mutations done in physical proteins. We identified 10 wild type, mutant pairs in the PDB, and *in silico* mutated the wild type structures to have the amino acid substitutions of the mutants in the PDB. As we did when assessing the quality of performing single *in silico* mutations, we calculated the all-atom RMSD values between the actual and generated mutant files. We also tallied the run-times of both FoldX and **ProMuteHT** (Table 4). The RMSD values all are below 1Å, and indeed for several structures with mutations at residues that are buried, the RMSD alignment between our generated and actual mutant structures were below 0.5Å. The run-times show that **ProtMuteHT** was faster (often twice as fast) than FoldX. We show in Figure 3 the alignment for the generated (from wild type 2lzm) and 190l (actual) mutant, both with 3 substitutions.

6 CONCLUSIONS

We have developed **ProMuteHT** for generating single and multiple amino acid substitutions *in silico*. With the aim of reducing compute times in the case that large numbers of mutants need to be generated, our goal from the onset was to limit the use of energy and force field calculations, especially when performing amino acid substitutions that did not produce steric clashes, which occurs when mutating from a large to a small residue. This approach has enabled us to achieve considerable speedup over FoldX, while still generating quality mutant structures. Our software also provides a simple command-line syntax to facilitate easily specifying which mutants to generate.

A limitation of our current implementation is mutating multiple amino acids in different chains of a multi-chain protein. We are prevented from currently doing this because of the choice of syntax for specifying which mutations should be *in silico* generated.

In on-going work, we are expanding and modifying the syntax of the input parameters to provide a user an even richer set of mutation options that they might specify. Also, we are extending the **ProMuteHT** software to leverage multiple cores to expedite the generation of mutants even more. Extensions to our software will also provide the user with options to specify parameters for the energy minimization portion of the compute pipeline. Lastly, the software is being deployed onto a web-server, which will provide an easy-to-use Graphical User Interface enabling generating many mutants with just a few mouse clicks. The **ProMuteHT** software is available from the authors for non-commercial use.

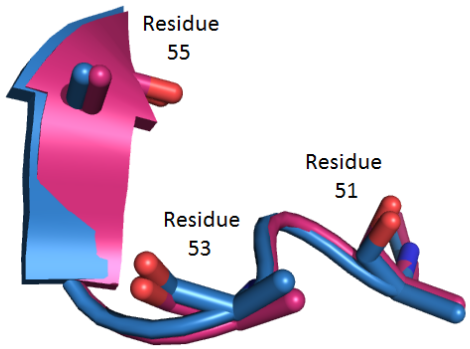

Figure 3: Aligning a mutant generated *in silico* by ProMuteHT (blue) with an actual mutant structure with equivalent mutations (magenta) shows a very good match. Amino acids substitutions performed were N53, N55A, and V57A in input PDB wild type structure 2lzm. The generated mutant achieved an RMSD of 0.20Å with the pdb file 190l which contains the actual mutations.

Table 4: Run-time and performance quality measurements for generating mutants with multiple amino acid substitutions. We used all-atom RMSD of the entire structure to measure the quality of the generated mutant when compared to the actual PDB mutant structure with equivalent mutations. Run-times are in seconds. SS=Secondary Structure, and ASA=Accessible Surface Area.

PDB WT	Mutant	Mutations	SS	ASA	all-atom RMSD	Run-times (s)	
						FoldX	ProMuteHT
1LZ1	1LHM	C 77 A, C 95 A	T, H	14.67, 1.28	0.203	3.4	1.7
4LYZ	1HEN	I 55 V, S 91 T	T, H	0.43, 0.43	0.407	3.1	1.3
4LYZ	1HEQ	T 40 S, S 91 T	S, H	0.07, 0.43	0.416	3.2	1.6
2LZM	1L49	A 98 V, T 152 S	H, H	0.00, 0.00	0.211	5.1	1.9
2LZM	1L89	L 99 A, F 153 A	H, H	0.00, 0.05	0.250	7.1	2.6
4LYZ	1HEP	T 40 S, I 55 V, S 91 T	S, T, H	0.07, 0.43, 0.43	0.424	6.7	2.5
2LZM	190L	N 53 A, N 55 A, V 57 A	C, T, S	85.72, 78.55, 44.24	0.202	7.3	2.4
2LZM	1L50	A 98 V, V 149 C, T 152 S	H, H, H	0.00, 0.07, 0.00	0.195	6.8	2.2
2LZM	1L51	A 98 V, V 149 I, R 152 S	H, H, H	0.00, 0.07, 0.00	0.225	7.2	2.3
451C	1DVV	F 7 A, V 13 M, F 34 Y, E 43 Y, V 78 I	T, T, T, T, H	1.89, 54.33, 39.66, 59.42, 6.90	0.869	9.6	3.1

REFERENCES

[1] Pedro J Ballester and John BO Mitchell. 2010. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics* 26, 9 (2010), 1169–1175.

[2] Jeffrey A Bell, Wayne J Becktel, Uwe Sauer, Walter A Baase, and Brian W Matthews. 1992. Dissection of helix capping in T4 lysozyme by structural and thermodynamic analysis of six amino acid substitutions at Thr 59. *Biochemistry* 31, 14 (1992), 3590–3596.

[3] Frances C Bernstein, Thomas F Koetzle, Graheme JB Williams, Edgar F Meyer, Michael D Brice, John R Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. 1977. The protein data bank. *European Journal of Biochemistry* 80, 2 (1977), 319–324.

[4] M J Bower, F E Cohen, and R L Jr Dunbrack. 1997. Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: a new homology modeling tool. *J Mol Biol* 267, 5 (1997), 1268–1282.

[5] Jeffrey R Brender and Yang Zhang. 2015. Predicting the effect of mutations on protein-protein binding interactions through structure-based interface profiles. *PLoS Comput Biol* 11, 10 (2015), e1004494.

[6] Yana Bromberg and Burkhard Rost. 2008. Comprehensive in silico mutagenesis highlights functionally important residues in proteins. *Bioinformatics* 24, 16 (2008), i207–i212.

[7] J. Cheng, A. Randall, and P. Baldi. 2006. Prediction of Protein Stability Changes for Single-Site Mutations Using Support Vector Machines. *PROTEINS: Structure, Function, and Bioinformatics* 62 (2006), 1125–1132.

[8] Warren L DeLano. 2002. The PyMOL molecular graphics system. (2002).

[9] R.L. Jr. Dunbrack and M. Karplus. 1994. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Structural Biology* 1 (1994), 334–340. Issue 5.

[10] D. Gilis and M. Rooman. 1997. Predicting Protein Stability Changes Upon Mutation Using Database-dervied Potentials: Solvent Accessibility Determines the Importance of Local Versus Non-Local Interactions Along the Sequence. *Journal of Molecular Biology* 272, 2 (1997), 276–290.

[11] Jeffrey J Gray, Stewart Moughon, Chu Wang, Ora Schueler-Furman, Brian Kuhlman, Carol A Rohl, and David Baker. 2003. Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of molecular biology* 331, 1 (2003), 281–299.

[12] Nicolas Guex and Manuel C Peitsch. 1997. SWISS-MODEL and the Swiss-Pdb Viewer: an environment for comparative protein modeling. *electrophoresis* 18, 15 (1997), 2714–2723.

[13] Maximilian Hecht, Yana Bromberg, and Burkhard Rost. 2015. Better prediction of functional effects for sequence variants. *BMC genomics* 16, 8 (2015), S1.

[14] F. Jagodzinski, B. Akbal-Delibas, and N. Haspel. 2013. An Evolutionary Conservation & Rigidity Analysis Machine Learning Approach for Detecting Critical Protein Residues. In *CSBW (Computational Structural Bioinformatics Workshop), in proc. of ACM-BCB (ACM International conference on Bioinformatics and Computational Biology)*. 780–786.

[15] J Janin and S Wodak. 1978. Conformation of amino acid side-chains in proteins. *J Mol Biol* 125 (1978), 357–386.

[16] Lei Jia, Ramya Yarlagadda, and Charles C Reed. 2015. Structure Based Thermostability Prediction Models for Protein Single Point Mutations with Machine Learning Tools. *PloS one* 10, 9 (2015), e0138022.

[17] Georgii G Krivov, Maxim V Shapovalov, and Roland L Dunbrack. 2009. Improved prediction of protein side-chain conformations with SCWRL4. *Proteins: Structure, Function, and Bioinformatics* 77, 4 (2009), 778–795.

[18] C. Lee and M. Levitt. 1991. Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. *Nature* 352 (1991), 448–451.

[19] Yunqi Li and Jianwen Fang. 2012. PROTS-RF: a robust model for predicting mutation-induced protein stability changes. *PloS one* 7, 10 (2012), e47247.

[20] Sushil Kumar Mishra, Jan Adam, Michaela Wimmerovaffi, and Jaroslav Kocffha. 2012. In silico mutagenesis and docking study of Ralstonia solanacearum RSL lectin: performance of docking software to predict saccharide binding. *Journal of chemical information and modeling* 52, 5 (2012), 1250–1261.

[21] Blaine HM Mooers, Walter A Baase, Jonathan W Wray, and Brian W Matthews. 2009. Contributions of all 20 amino acids at site 96 to the stability and structure of T4 lysozyme. *Protein Science* 18, 5 (2009), 871–880.

[22] James C Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D Skeel, Laxmikant Kale, and Klaus Schulten. 2005. Scalable molecular dynamics with NAMD. *Journal of computational chemistry* 26, 16 (2005), 1781–1802.

[23] J.W. Ponder and F.M. Richards. 1987. Tertiary templates for proteins: Use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal Molecular Biology* 193 (1987), 775–791. Issue 4.

[24] Boris Reva, Yevgeniy Antipin, and Chris Sander. 2011. Predicting the functional impact of protein mutations: application to cancer genomics. *Nucleic Acids Research* (2011).

[25] John A Schellman. 1987. The thermodynamic stability of proteins. *Annual review of biophysics and biophysical chemistry* 16, 1 (1987), 115–137.

[26] Joost Schymkowitz, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. 2005. The FoldX web server: an online force field. *Nucleic acids research* 33, suppl 2 (2005), W382–W388.

[27] C.M. Topham, N. Srinivasan, and T. Blundell. 1997. Prediction of the stability of protein mutants based on structural environment-dependent amino acid substitutions and propensity tables. *Protein Engineering* 10, 1 (1997), 7–21.

[28] C.L. Worth, R. Preissner, and L. Blundell. 2011. SDM-a server for predicting effects of mutations on protein stability and malfunction. *Nucleic Acids Research* 39, Web Server Issue (2011), W215–W222.

[29] Lingchong You and John Yin. 2002. Dependence of epistasis on environment and mutation severity as revealed by in silico mutagenesis of phage T7. *Genetics* 160, 4 (2002), 1273–1281.