A Bioinformatics Approach to Increasing the Solubility of Protein-Based Drugs

Aamod S. Dekhne

April 2014

Advisor:

Dr. David Shultis

Yang Zhang Lab

Department of Computational Medicine and Bioinformatics

University of Michigan

Abstract

This thesis presents the construction and application of a pipeline combining the protein structure prediction utilities ThreaDom, I-TASSER, and STRIDE. ThreaDom predicts the residue boundaries of each domain in a protein by multiple threading alignments. I-TASSER analyzes the amino acid sequence of a protein through iterative threading and refinement of decoys to generate a predicted three-dimensional structure. STRIDE determines secondary structure from the I-TASSER model to calculate the solvent accessibility of each residue. The programs are bridged together by scripts written in Perl to sequentially determine the domains of an input protein and then to generate predicted structures of the individual domains. Another script then applies Kyte-Doolittle hydrophobicity scores and STRIDE solvent accessibility calculations to determine exposed hydrophobic residues. Nearest neighbor analysis of those exposed hydrophobic residues then determines high-risk hydrophobic patches which could potentially lead to unwanted aggregation in novel protein-based drug compounds. The BLOSUM-62 substitution matrix could then be used to generate an alternative protein sequence that would decrease aggregation while maintaining the structure and function of the original, thus increasing the bioavailability and effectiveness of the protein-based drug.

Contents

Chapter 1

Introduction

Protein-based therapeutic drugs have demonstrated great success in clinical studies and account for over $100 billion in US sales alone. These proteins can act by bolstering deficient proteins, interfering with existing molecules or pathogens, or providing a novel function or activity [6]. As a protein's function is derived purely from its three dimensional structure, proteins with novel functions must also have novel structures which may or may not be stable or soluble. Currently, most *de novo* protein structure design projects rely on heavy experimental testing of the protein's stability, which is highly time consuming and labor intensive [7]. The alternative to experimental screening of the created protein is to attempt to forecast the protein's stability before it is created.

While a protein-based drug's efficacy is determined by many different characteristics of its structure, this thesis focuses on forecasting the tendency of a novel protein-based drug to undergo hydrophobic aggregation which in turn would limit the drug's solubility, bioavailability, and ultimately its efficacy. The root cause of hydrophobic aggregation is the presence of solvent exposed patches of hydrophobic residues as shown below in Figure 1.

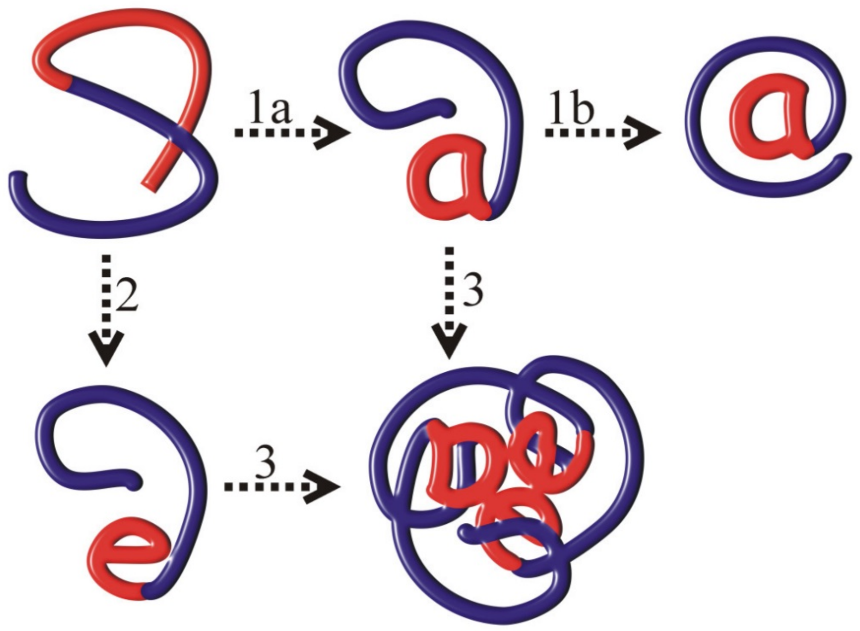


Figure 1: Hydrophobic protein aggregation. The red and blue regions of the protein represent hydrophobic and hydrophilic regions of the protein respectively. Pathway 1a→1b demonstrates the correct folding leading to a functional, soluble protein wherein the hydrophobic region is solvent-protected and does not aggregate. Pathway 2→3 demonstrates improper folding resulting in a large solvent-exposed hydrophobic patch and subsequent aggregation. From [7]

With the availability of individual software packages to predict individual protein domains from the primary structure [1], predict the structure of each of these domains [2], and determine the solvent accessibility and hydrophobicity of each residue in a protein structure [3], it is possible to combine all of these programs into one pipeline. The pipeline, then, would fold an inputted protein-based drug's amino acid sequence and determine the regions of the protein most susceptible to causing aggregation. It could also be expanded to incorporate BLOSUM-62, a matrix that numerically defines the extent to which a given residue point mutation is favorable or unfavorable [5]. BLOSUM-62 could be used to propose a new sequence that would maximize the drug's solubility by removing hydrophobic patches while minimizing unfavorable mutations and avoiding changes to the drug's overall structure and function.

Chapter 2

Pipeline Construction

2.1 Pipeline Architecture

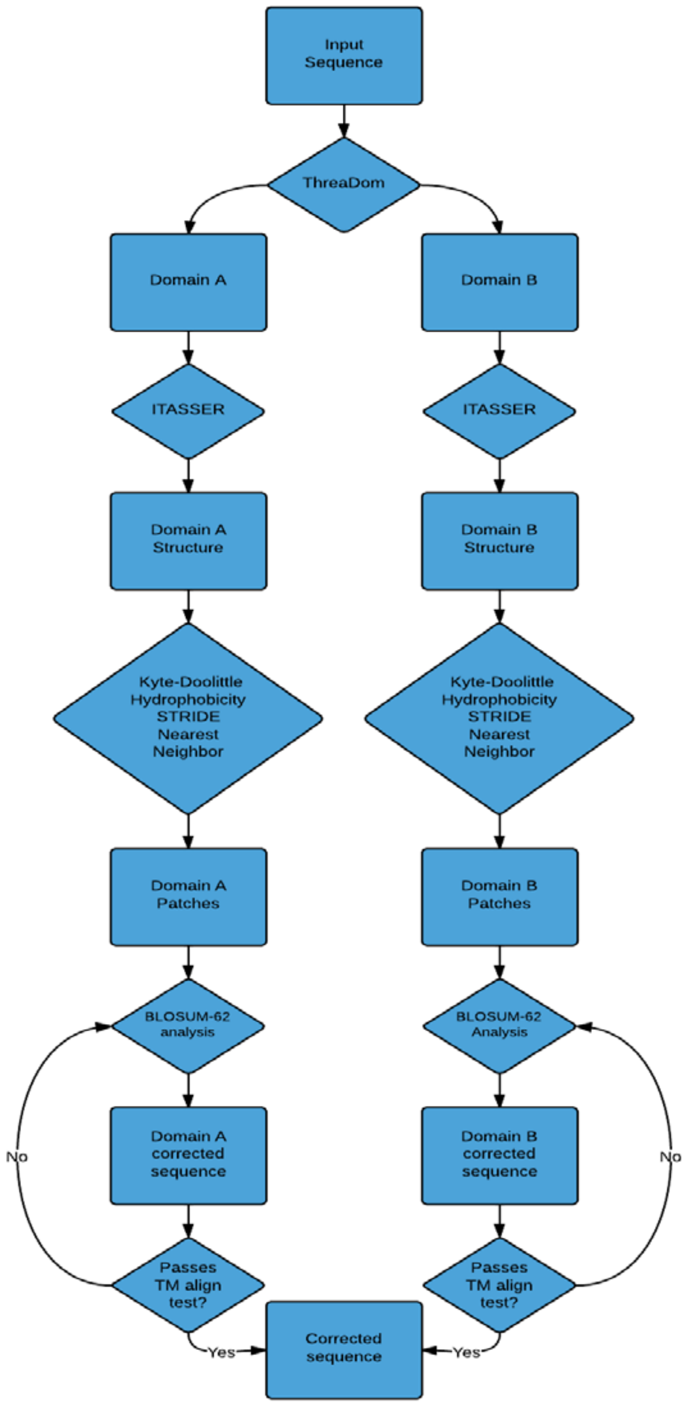


Figure 2: Architecture of the pipeline. It runs the input amino acid sequence through ThreaDom to find the domain boundaries. Then ITASSER generates a PDB file with the predicted coordinates of each atom in each domain. A script then analyzes the hydrophobicity, STRIDE solvent-accessibility, and nearest neighbors to determine the exposed hydrophobic patches. BLOSUM-62 could then be used to suggest changes to the sequence to increase solubility. The corrected sequences would then be checked for functional fidelity by TM align and either remade or combined into a final corrected sequence.

2.2 Data Sets

The training set consists of 3527 proteins taken at random from the Protein Data Bank (PDB).

2.3 ThreaDom

A script (see Appendix: A) allows the user to define the parameters for each individual program in one centralized location. A modified ThreaDom program (see Appendix: B) then iterates through each of the proteins in the training set. ThreaDom takes a primary protein sequence as input and outputs the residue numbers of predicted domain boundaries as shown in figure 2 below [1]. Each domain can then be analyzed separately for function and solubility.

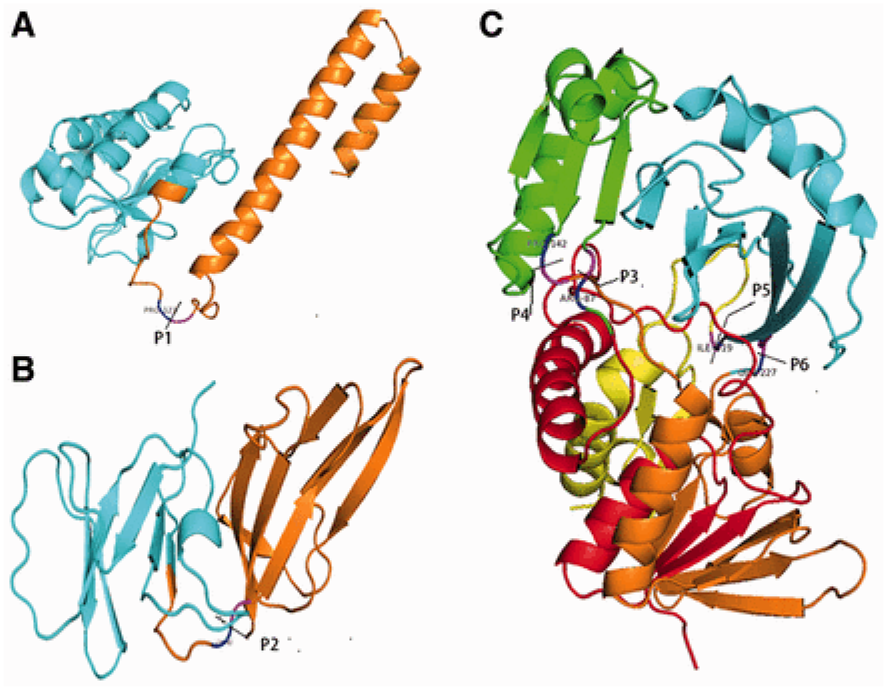


Figure 3: Visual examples of ThreaDom domain prediction. ThreaDom predicted domains segments are denoted by different colors. From [1].

2.4 ITASSER

After ThreaDom outputs the predicted domains, another script (see Appendix: C) extracts the boundaries and divides the original sequence into domain-specific inputs for ITASSER. ITASSER takes the primary sequence of each ThreaDom-predicted domain as input and, through **I**terative **T**hreading**, Asse**mbly, and **R**efinement (as shown below in figure 4), generates a PDB file with the predicted coordinates of each atom in the protein [2].

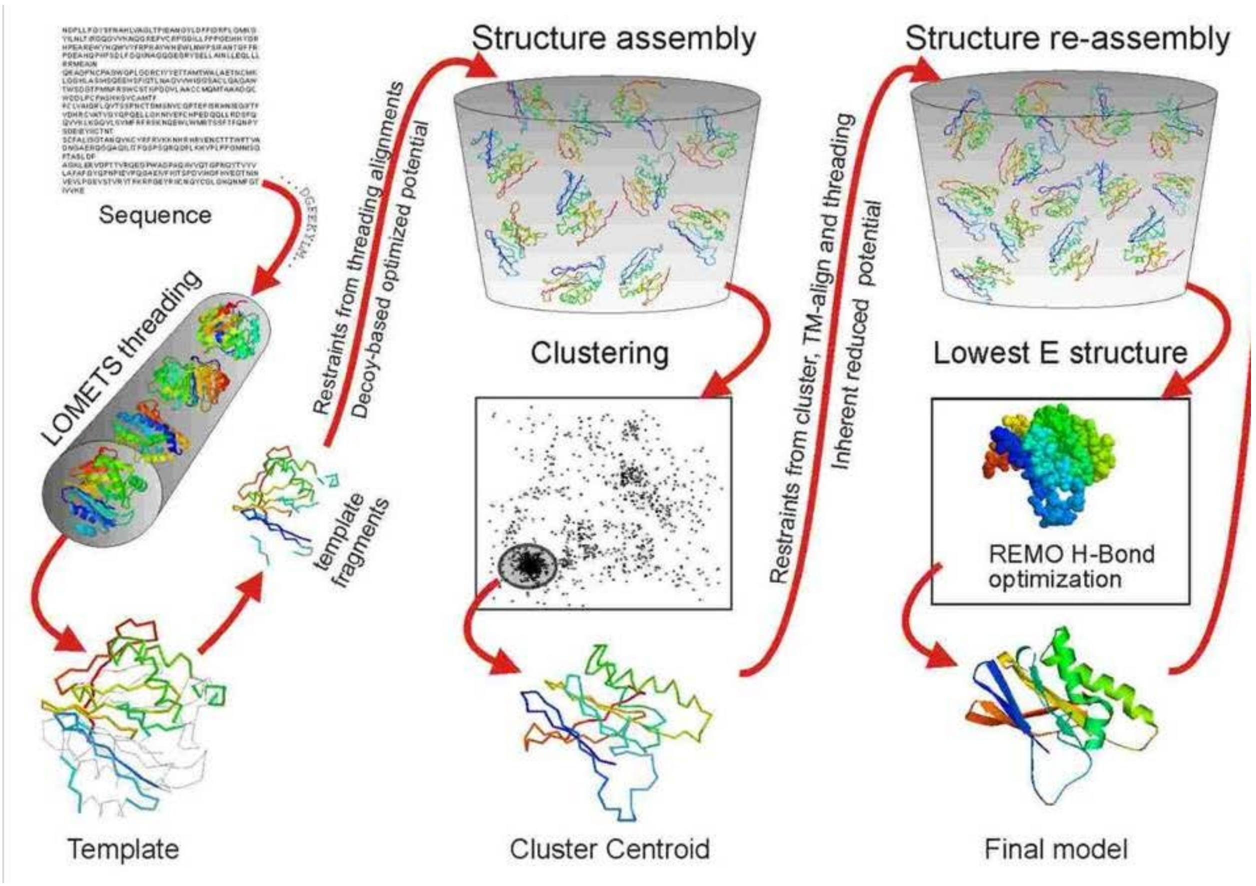


Figure 4: Architecture of ITASSER. The primary sequence is threaded through the PDB library to identify possible templates of known structure on which to base the final model. Threading applies sequence profile alignments and secondary structure comparison to determine templates that resemble the input sequence structure. These templates fragments are then assembled and the cluster centroid (i.e. the structure which most closely resembles each of the other predicted structures) is further optimized through hydrogen bond optimization and steric clash removal to generate a final PDB model. From [2]

2.5 STRIDE

The ITASSER predicted models must then be analyzed for solvent accessibility. Interior (i.e. solvent-protected) hydrophobic patches are necessary for proper folding and function of the protein and so, should not be altered. STRIDE (**Str**uctural **Ide**ntification) takes the ITASSER predicted coordinates as input and applies the double cubic lattice method to calculate the solvent-exposed area of each residue in the protein [9]. The solvent-accessible area of each residue is then divided by the total accessible surface area of that residue (as calculated in [10]) to yield a percent solvent accessibility. A residue with accessibility greater than or equal to 50% is classified as "solvent-exposed". Any residues with less than 50% accessibility are classified as "buried" in the hydrophobic core and are excluded from mutation in order to maintain the stability of the protein.

2.6 Kyte-Doolittle Hydrophobicity

Of the solvent-exposed residues as defined by STRIDE, those that are also hydrophobic may lead to aggregation. The Kyte-Doolittle hydrophobicity scale is a means of quantifying each residue's hydrophobicity from -4.5 for the least hydrophobic residue arginine to +4.5 for the most hydrophobic residue isoleucine [11]. The hydrophobicity values were derived primarily experimentally from the water-vapor transfer free energies and interior-exterior distribution of each residue. These initial values were then adjusted subjectively where the authors deemed that the structure of a given residue merited a change in the residue's experimental ranking. Ultimately, a moving average plot of the Kyte-Doolittle hydrophobicity over 19 residues (as shown below in figure 5) yielded a cutoff value of 1.6. Any regions in the hydropathy plot above 1.6 have a high probability of being an intra-membrane protein and 1.6 was initially used as the cutoff for defining a residue as hydrophobic. However, a value of 1.6 proved too restrictive in initial testing and the cutoff point was redefined to 0.0 in the pipeline.

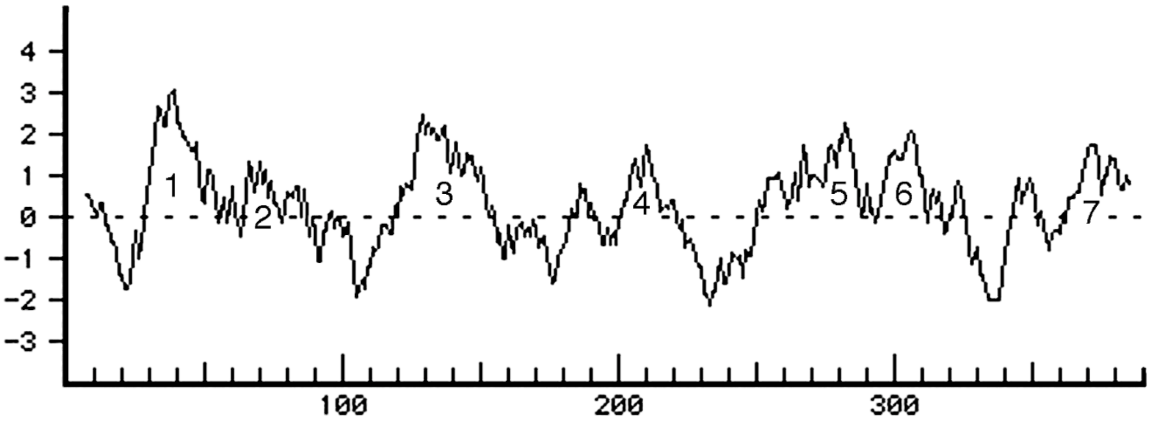


Figure 5: Hydropathy plot (Kyte-Doolittle moving average versus residue number) of an odorant receptor (RSOr1) in the Northern Walnut Husk Fly, *Rhagoletis suavis*. From [11]

2.7 Nearest-Neighbor Analysis

The Kyte-Doolittle moving average works linearly through the primary sequence of the protein. However, hydrophobic aggregation is not caused by many consecutive hydrophobic residues, but rather by many spatially tightly-packed hydrophobic residues. Another script (see Appendix: D) pulls the alpha carbon coordinates of each residue from the ITASSER output and calculates every solvent-exposed residue within an 8 angstrom radius (nearest neighbors). Each residue with more than three hydrophobic, solvent-exposed neighbors qualifies as a starting point for determining a hydrophobic patch (patch-seed). Each patch-seed is analyzed by the same script according to the flowchart in figure 6. After each patch-seed's patch has been determined, the repeated patches (which stem from the fact that two patch-seed residues can be in the same patch) are excised along with any patches with fewer than 3 residues (which represent isolated hydrophobic areas). Finally, the script outputs a final list of unique, solvent-exposed, hydrophobic patches in the protein and assigns each patch a score based on the patch's size and average hydrophobicity.

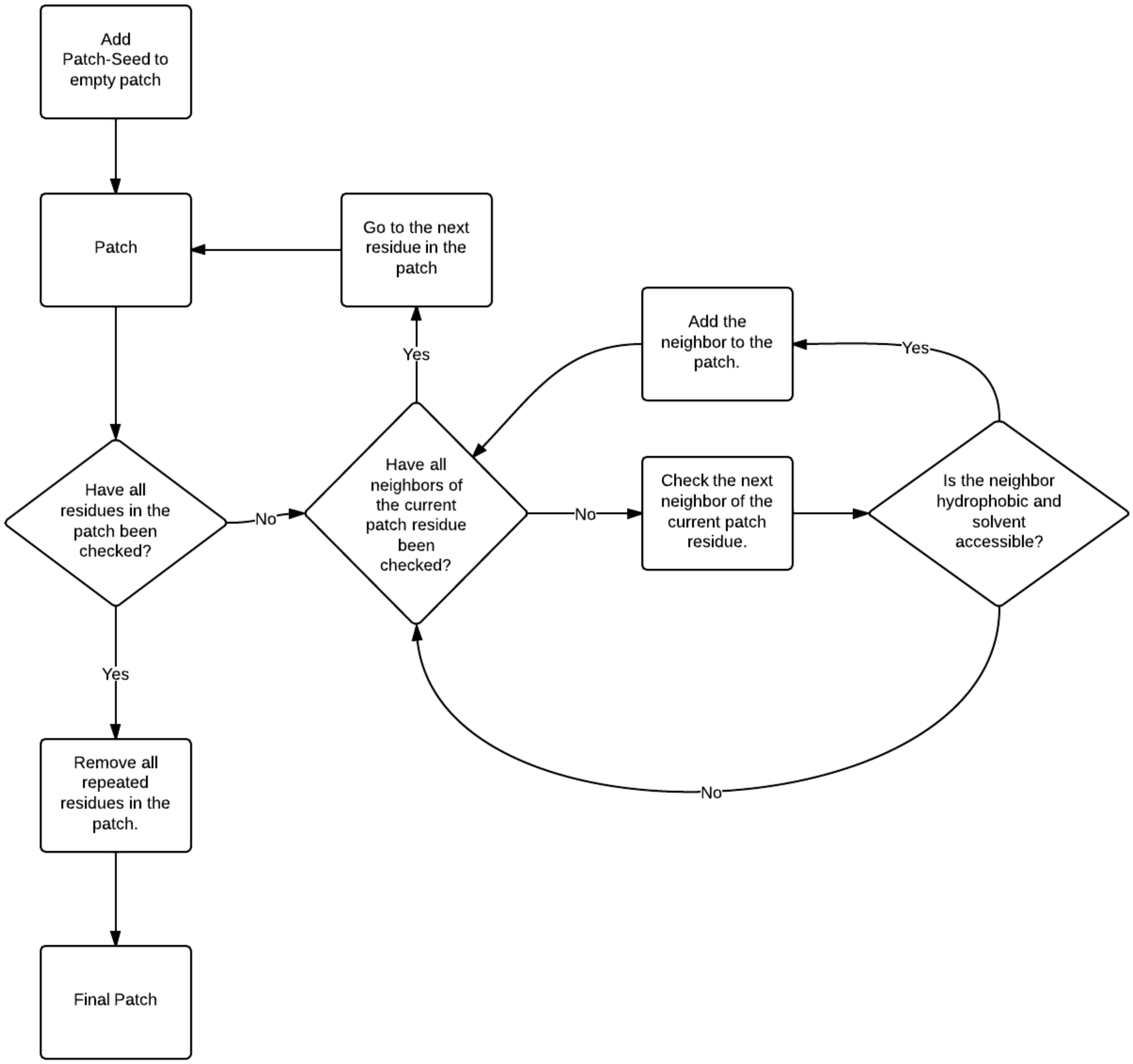
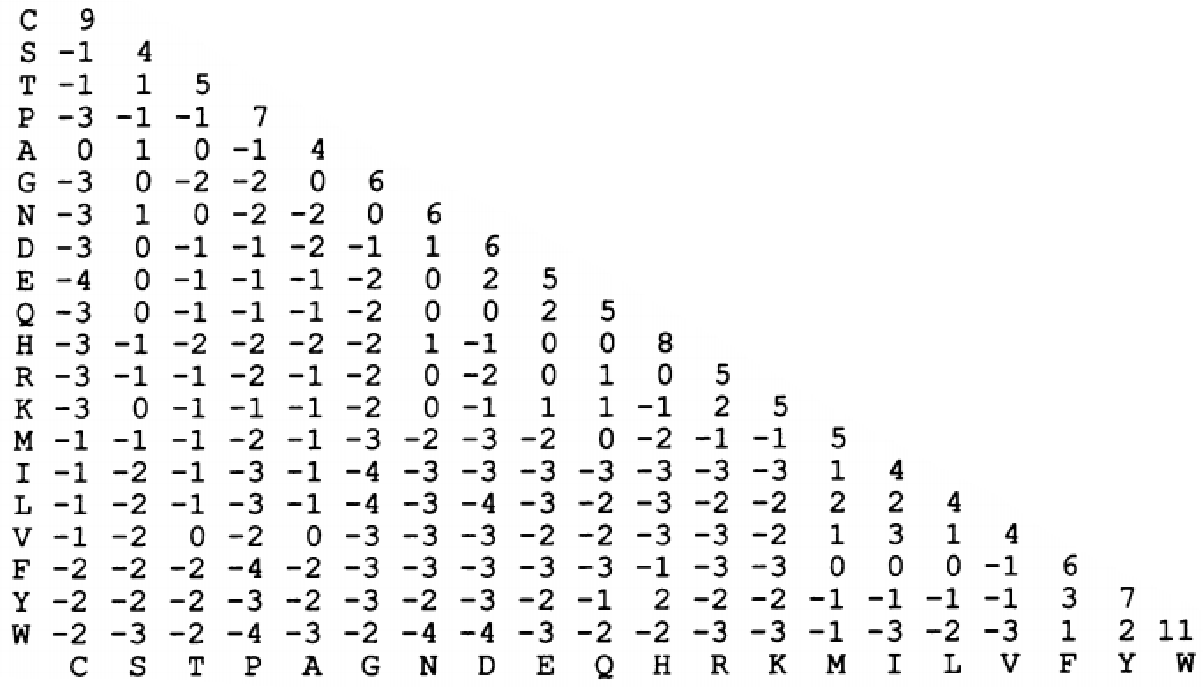

Add Patch-Seed to empty patch
Patch
Go to the next residue in the patch
Add the neighbor to the patch.
Yes
Yes
Have all residues in the patch been checked?
No
Have all neighbors of the current patch residue been checked?
No
Check the next neighbor of the current patch residue.
Is the neighbor hydrophobic and solvent accessible?
Yes
No
Remove all repeated residues in the patch.
Final Patch

Figure 6: The first patch-seed is added to the patch. Then the script checks the neighbors of each residue in the patch for hydrophobicity and solvent-accessibility. If the neighbor passes the check, it is added to the patch and its own neighbors in turn are analyzed. When all of the residues stemming from a given patch-seed have been analyzed, the repeat residues are removed and the patch then consists of unique residues.

2.8 BLOSUM-62

After the final list of hydrophobic patches is generated, those patches necessary for protein function (such as dimer interfaces) should not be altered. The remaining patches should be altered with minimum impact on function. One method of determining residue mutations with no functional impact on the protein is to study the rate at which a given residue changes to another residue in evolutionarily homologous sequences. The **Blo**cks **Su**bstitution **M**atrix with alignment threshold of **62%** (see figure 7 below) is one such tool that clusters sequences with sequence identity greater than 62% and compares the changes in residues among clusters. Each pair of residues in the matrix is assigned a mutation score. A higher score signifies a more favorable mutation than a lower score.



C	9																			
S	-1	4																		
T	-1	1	5																	
P	-3	-1	-1	7																
A	0	1	0	-1	4															
G	-3	0	-2	-2	0	6														
N	-3	1	0	-2	-2	0	6													
D	-3	0	-1	-1	-2	-1	1	6												
E	-4	0	-1	-1	-1	-2	0	2	5											
Q	-3	0	-1	-1	-1	-2	0	0	2	5										
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8									
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5								
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5							
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5						
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4					
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4				
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4			
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Figure 7: BLOSUM-62 matrix. A higher pairwise score indicates favorable substitution and a lower score indicates unfavorable substitution. From [5]

2.9 TM-Align

After a script analyzes the hydrophobic patches for the residue substitutions that would result in the most favorable BLOSUM-62 substitution and greatest decrease in hydrophobicity, the resulting corrected sequence must be checked for functional fidelity. TM-Align is a structure alignment program that generates a score between 0.0 and 1.0 based on the similarity between two structures as seen in figure 8 [11]. A cutoff score of 0.5 was used to define distinct folds. The corrected sequence could be re-processed through ThreaDom and ITASSER to generate a corrected structure. TM-align can then compare the original structure and corrected structure. If the comparison yields a score under the cutoff of 0.5, the script can redo the BLOSUM-62 analysis with the next most favorable residue substitution. The process can be repeated until a TM-score of greater than 0.5 is achieved or no possible substitutions exist. Ultimately, a sequence which generates a structure with TM-align score greater than 0.5 would be output as the final corrected sequence.

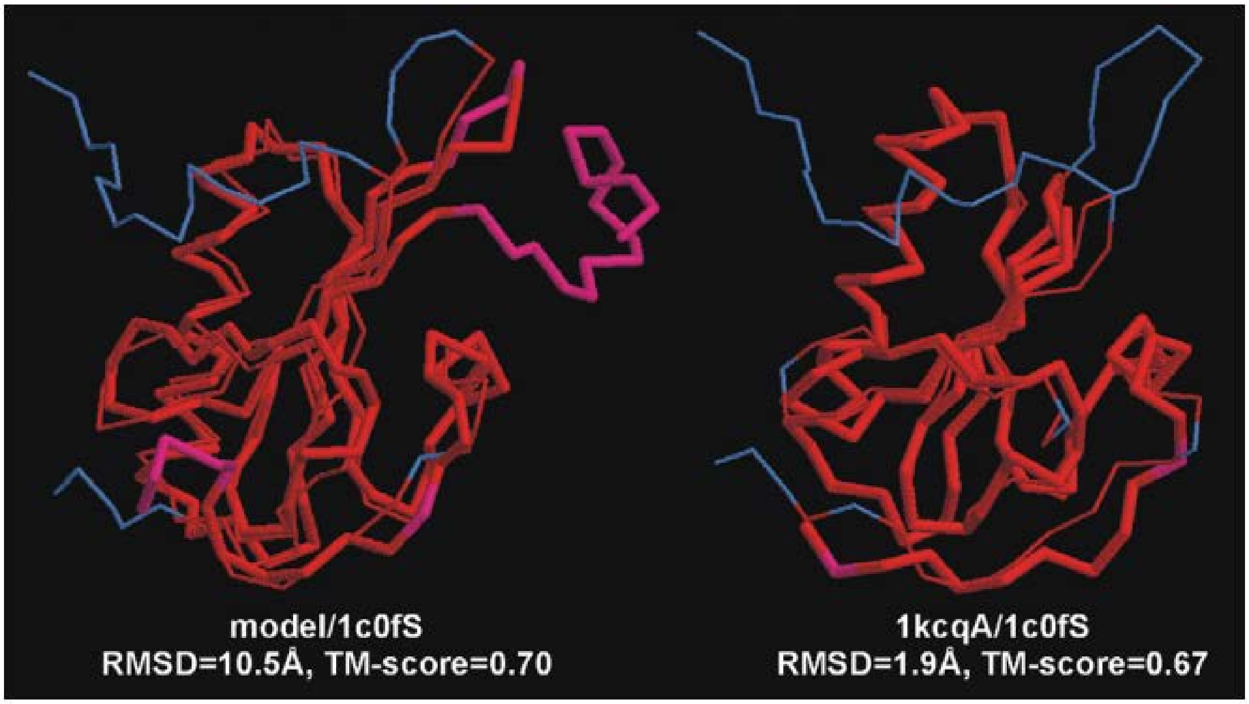


Figure 8: Sample TM-align comparisons. The red regions are residues within 5 angstroms of the original structure. From [11].

Chapter 3

Results

The pipeline has been constructed up until BLOSUM-62 analysis. Therefore, the present output is a PDB that identifies the hydrophobic patches in an input protein sequence as shown below in figure 9.

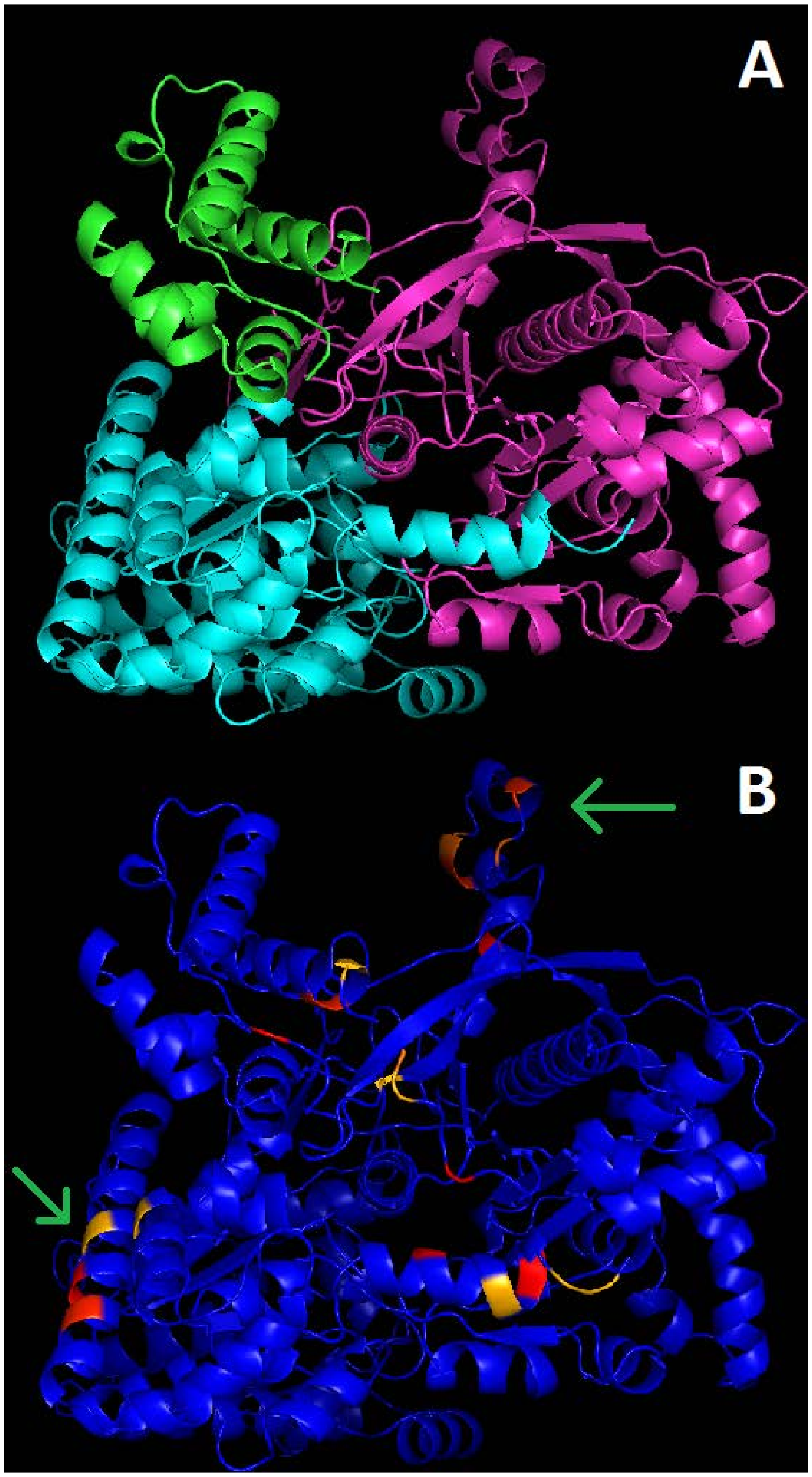


Figure 9: Two PyMol representations of 1ahp, an E. Coli maltodextrin phosphorylase. Figure 9A shows the enzyme with the 3 domains in different colors. Figure 9B shows the same enzyme with the hydrophobic patches indicated in red and yellow. Note that the majority of these hydrophobic patches occur at domain interfaces. Patches that do not occur at known interfaces and may lead to unintended aggregation are noted with green arrows.

Chapter 4

Conclusion

At a minimum, the pipeline should identify dimer-dimer and domain interfaces as aggregation of proteins at these locations is critical to protein function. As shown in figure 9, of the six hydrophobic patches detected, four occur at domain interfaces. These patches will not be altered to preserve proper function. The two patches indicated by the green arrows do not occur at domain or dimer interfaces. As the protein 1ahp is an enzyme, it is possible that one or both of these sites serves as the binding location for the substrate and so, should also not be altered. Another existing program, COACH, can predict protein-ligand binding sites [12]. After the completion of the pipeline with BLOSUM-62 analysis, future direction for expansion of the pipeline's function may include incorporation of COACH to avoid changing the protein's active site. Ultimately, this pipeline will help pave the way for more effective protein-based therapeutics.

Chapter 5
References

1. Xue Z, Xu D, Wang Y, Zhang Z. ThreaDom: Extracting Protein Domain Boundary Information from Multiple Threading Alignments. *Bioinformatics*, 29: i247-i256. (2013)

2. Roy, A, Kucukural, A Zhang, Y. I-TASSER: a unified platform for automated protein structure and function prediction. *Nature Protocols*, vol 5, 725-738. (2010)

3. Heinig, M., Frishman, D.. STRIDE: a Web server for secondary structure assignment from known atomic coordinates of proteins. *Nucl. Acids Res.* 32, W500-2. (2004)

4. Kyte, J. and Doolittle, R. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.* 157: 105-132. (1982)

5. Henikoff, S.; Henikoff, J.G. Amino Acid Substitution Matrices from Protein Blocks. *PNAS* **89** (22): 10915–10919. (1992)

6. Dimitrov DS. Therapeutic Proteins. *Methods Mol Biol.*; 899:1-26. (2012)

7. Zhixiu L et al. Energy Functions in De Novo Protein Design: Current Challenges and Future Prospects *Annual Review of Biophysics*. 42: 315-335 (2013)

8. Vallejo LF, Rinas U. Strategies for the recovery of active proteins through refolding of bacterial inclusion body proteins *Microbial Cell Factories*. 3:11 (2004)

9. Eisenhaber, F et al., The double cubic lattice method: Efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies. *J. Comput. Chem.*, 16: 273–284. (1995)

10. Miller et al. Interior and Surface of Monomeric Proteins *J. Mol. Biol.* 196, 641-656 (1987)

11. Zhang, Y. Skolnick, J. TM-Align: A protein structure alignment algorithm based on TM-score, *Nucleic Acids Res.*, 33: 2302-2309 (2005)

12. Yang, J., Roy, A, Zhang, Y. Protein-ligand binding site recognition using complementary binding-specific substructure comparison and sequence profile alignment, *Bioinformatics*, 29:2588-2595 (2013)

Chapter 6

Appendices

6.1 Appendix A: Centralized Variable Definitions

```
#!/usr/bin/perl
use warnings;
use strict;
use Cwd;
#This program needs a folder with a protein fasta list
#This program needs a protein list with the name of the folder
#This needs an isreal.txt file set ot 1 or 0
#This needs a status.txt file set to nothing

#if you are running threadom from someone else's directory, change these variables and those
below
my $pwd;
$pwd = getcwd(); #defines base if being run in own directory
my $base = "$pwd/program"; #comment out $pwd and define if not accurate
my $outdir = "$pwd/newtest"; #location of protein.lst and sequence folders, same as outdir
my $user = `whoami`; #username
chop ($user);
my $bindir = "$base/itasser20101123"; #where threadom itasser script and cas programs are;
my $itasserbin = "$base/version_2013_03_20_f"; #where itasser script and cas programs are;
my $itasserout = "$pwd/newtestout"; #where itasser output is;
my $stridebin = "$pwd/program"; #where stride exe is

#if you are running threadom set up in your own directory, only change these variables
my $njobmax = 300; #max jobs submitted by you
my $njoballmax = 2000; #max jobs submitted by whole lab;
my $Q = "default"; #queue
my @proteinlist = ("sodium", "mumps", "hydrophobic"); #protein names separated by commas in
quotes eg ("protein1", "protein2")
my $benchmark = "real"; #set to "real" or "benchmark";
my $option = '1'; #1 for combo models. 2 for closc models;
my $hpthreshold = 1.6; #hydrophobicity score threshold for hydrophobic residue (default 1.6)
### END OF NECESSARY CHANGE


#create temporary module in $base/
my $filename = "$base/tempmodule/tempmodule.pm";
unless(open FILE, '>', $filename) {
	die "\nUnable to create $filename\n";
}
```

```
print "tempmodule created\n"; sleep (1);
print FILE '#!/usr/bin/perl
use warnings;
use strict;

package';
print FILE " tempmodule;";
print FILE '

use Exporter;
our @ISA = \'Exporter\';
our @EXPORT = qw(
        $base
        $outdir
        $user
        $stridebin
        $bindir
        $njobmax
        $njoballmax
        $Q
        @proteinlist
        $itasserbin
        $itasserout
        $benchmark
        $option
        %hydrophobic
        $hpthreshold
        %basesolventaccessibility
);



our $base; our $outdir; our $user; our $stridebin; our $option; our $bindir; our $njobmax; our
$benchmark; our $njoballmax; our $Q; our @proteinlist; our $itasserbin; our $itasserout;
our %hydrophobic; our $hpthreshold; our %basesolventaccessibility;

$base = '; print FILE "\"$base\"\;\n";
print FILE '$benchmark ='; print FILE "\"$benchmark\"\;\n";
print FILE '$outdir = '; print FILE "\"$outdir\"\;\n";
print FILE '$user = '; print FILE "\"$user\"\;\n";
print FILE '$stridebin = '; print FILE "\"$stridebin\"\;\n";
print FILE '$bindir = '; print FILE "\"$bindir\"\;\n";
print FILE '$njobmax = '; print FILE "\"$njobmax\"\;\n";
print FILE '$njoballmax = '; print FILE "\"$njoballmax\"\;\n";
print FILE '$Q = '; print FILE "\"$Q\"\;\n";
print FILE '@proteinlist = qw('; foreach(@proteinlist) {print FILE "$_ ";}  print FILE ');'; print
FILE "\n";
print FILE '$itasserbin = '; print FILE "\"$itasserbin\"\;\n";
print FILE '$itasserout = '; print FILE "\"$itasserout\"\;\n";
print FILE '$option = '; print FILE "\"$option\"\;\n";

print FILE '%hydrophobic = '; print FILE "(
        I => 4.5, ILE => 4.5,
        V => 4.2, VAL => 4.2,
        L => 3.8, LEU => 3.8,
        F => 2.8, PHE => 2.8,
        C => 2.5, CYS => 2.5,
        M => 1.9, MET => 1.9,
        A => 1.8, ALA => 1.8,
        G => -0.4, GLY => -0.4,
        T => -0.7, THR => -0.7,
        W => -0.9, TRP => -0.9,
        S => -0.8, SER => -0.8,
        Y => -1.3, TYR => -1.3,
        P => -1.6, PRO => -1.6,
        H => -3.2, HIS => -3.2,
        E => -3.5, GLU => -3.5,
        Q => -3.5, GLN => -3.5,
        D => -3.5, ASP => -3.5,
        D => -3.5,
        N => -3.5, ASN => -3.5,
        K => -3.9, LYS => -3.9,
        R => -4.5, ARG => -4.5
);";

print FILE '%basesolventaccessibility = '; print FILE "(
        ILE => 147.3,
        VAL => 122.3,
        LEU => 146.2,
        PHE => 180.1,
        CYS => 102.3,
        MET => 158.3,
        ALA => 64.9,
        GLY => 87.2,
        THR => 106.2,
        TRP => 224.6,
        SER => 77.4,
        TYR => 193.1,
        PRO => 105.2,
        HIS => 154.6,
        GLU => 141.2,
        GLN => 143.7,
        ASP => 113.0,
        ASN => 114.3,
        LYS => 164.5,
        ARG => 195.5
);";

print FILE '$hpthreshold = '; print FILE "$hpthreshold\;\n";
print FILE '1;';


close FILE;

$pwd = getcwd();
`cp -fr "$base/tempmodule/tempmodule.pm" "$pwd"`; #for TH_round1 and TH_round2
`cp -fr "$base/tempmodule/tempmodule.pm" "$base/bin"`; #for Threadom programs
`cp -fr "$base/tempmodule/tempmodule.pm" "$bindir"`; #for threadom mkinput1
`cp -fr "$base/tempmodule/tempmodule.pm" "$itasserbin"`; #for itasser
my $myproteinlist = "$outdir/protein.lst";
unless (open PROTEINS, '>', $myproteinlist) {
        `rm -f "$base/tempmodule/tempmodule.pm"`;
        die "\nUnable to open protein.lst";
}
foreach (@proteinlist) {
print PROTEINS "$_\n";
}
close PROTEINS;

print "protein.lst created\n";
```

6.2 Appendix B: ThreaDom scripts

```
#!/usr/bin/perl


use strict;
use warnings;
require tempmodule;

#my $base        = "/home/adekhne/bin/threadom/program";
my $base = $tempmodule::base;
my $itasser_bin = "$base/itasser20101123";
my $dom_bin     = "$base/bin";
#my $outdir      = "$base/output";
my $outdir      = $tempmodule::outdir;
my @filelist;

open(LOOPLIST, "<$outdir/protein.lst");
@filelist=<LOOPLIST>;
chomp @filelist;
close(LOOPLIST);

foreach my $s(@filelist) {
    open(MLOG, ">$outdir/$s/masterlog.txt");

    #change job status to job_waiting
    `echo job_waiting > "$outdir/$s/status.txt"`;

   #RUN Mkinput 1

    my $isreal=`cat $outdir/$s/isreal.txt`;
   chop($isreal);
    print MLOG "Starting mkinput1\n";
    system("perl $itasser_bin/mkinput1.pl $s $isreal");
    `rm -f "$outdir/$s/status.txt"`;
    `echo job_running > "$outdir/$s/status.txt"`

}

close (MLOG);
exit();

#!/usr/bin/perl

# This program needs a folder with a protein fasta list
# This program needs a protein list with the name of the folder in it
# This needs an isreal.txt file seto to 1 or 0
# This needs an status.txt file set it to nothing
require tempmodule;

#my $base       = "/home/adekhne/bin/threadom/program";
my $base        = $tempmodule::base;
my $itasser_bin = "$base/itasser20101123";
my $dom_bin     = "$base/bin";
#my $outdir      = "$base/output";
my $outdir      = $tempmodule::outdir;
my @filelist;

open(LOOPLIST, "<$outdir/protein.lst");
@filelist=<LOOPLIST>;
chomp @filelist;
close(LOOPLIST);
`rm -rf "$base/output"`;
`cp -rf "$outdir" "$base/output"`;

foreach my $s(@filelist) {
   open(MLOG, ">$outdir/$s/masterlog_templates_Threadom.txt");
      `perl "$dom_bin/collectTemplates.pl" $s`;
      `perl "$dom_bin/ThreaDom.pl" $s`;
         `cp -rf "$base/output/$s" "$outdir"`;
         `perl "$dom_bin/draw.pl" $s`;
         `perl "$dom_bin/detectdcdv4_web.pl" $s`;
   }

close(MLOG);
open(LOOPLIST, "<$outdir/protein.lst");
@filelist=<LOOPLIST>;
chomp @filelist;
close(LOOPLIST);
`rm -rf "$base/output"`;
`cp -rf "$outdir" "$base/output"`;

foreach my $s(@filelist) {
   open(MLOG, ">$outdir/$s/masterlog_templates_Threadom.txt");
      `perl "$dom_bin/collectTemplates.pl" $s`;
      `perl "$dom_bin/ThreaDom.pl" $s`;
      `cp -rf "$base/output/$s" "$outdir"`;
      `perl "$dom_bin/draw.pl" $s`;
      `perl "$dom_bin/detectdcdv4_web.pl" $s`;
   }
close(MLOG);
`rm -f "$base/tempmodule/tempmodule.pm"`;
`rm -rf "$base/output"`;

exit();
```

6.3 Appendix C: Preparation for ITASSER

```
#!/usr/bin/perl
use warnings;
use strict;
use tempmodule;
use File::Path qw(mkpath);
my $numdom;
my $bound;
my $firstline;
my $fasta;
my $domseq;
my $dombound = 0;
my $hpbcount = 0;
my $domlength;

my %hydrophobic = (
        I => 4.5,
        V => 4.2,
        L => 3.8,
        F => 2.8,
        C => 2.5,
        M => 1.9,
        A => 1.8,
        G => -0.4,
        T => -0.7,
        W => -0.9,
        S => -0.8,
        Y => -1.3,
        P => -1.6,
        H => -3.2,
        E => -3.5,
        Q => -3.5,
        D => -3.5,
        N => -3.5,
        K => -3.9,
        R => -4.5,
            X => 0.0
);


#read in domains
foreach(@proteinlist) {
            print "------------"; print "$_"; print "------------"; print "\n";
            print "NUMBER OF DOMAINS : ";
            open INFILE,  "$outdir/$_/$_.result" or die "cannot open Threadom results";
            $firstline = <INFILE>;
            close INFILE;

            if ($firstline =~ /^\S+\s+\S+\s+(\d+)/) {$numdom = $1};
             print $numdom; print "\n";
            $bound="1";

            open FASTA, "$outdir/$_/$_.fasta" or die$!;
            $fasta = <FASTA>;
            close FASTA;

            for (my $domains = 1; $domains <= $numdom; $domains++) {
                    print "DOMAIN $domains : $bound \- ";
                    if ($firstline =~ /\($bound\-(\d+)\)/){
                            $bound = $1;
                             print "$bound\n";
                            $dombound = $bound - $dombound;
                            mkpath("$itasserout/$_-$domains");
                            open ITASSER, ">$itasserout/$_-$domains/seq.fasta" or die$!;
                            $domseq = substr($fasta, 0, $dombound, "");
                            print ITASSER "$domseq";
                            print "$_-$domains seq.fasta was created\n";
                            print "AVERAGE HYDROPHOBICITY : "; $hpbcount = 0; $domlength =
length($domseq);
                            for (my $key = 0; $key < $domlength ; $key++) {

                                    my $residue = substr($domseq, $key, 1);
                                    $hpbcount = $hydrophobic{$residue} + $hpbcount;
                            }
                            my $hpstatus = 0;
                            if ($hpbcount/$domlength > $hpthreshold) {$hpstatus = 1;}
                            printf("%.2f", $hpbcount/$domlength); if ($hpstatus) {print "\n"; print "$_-
$domains IS HYDROPHOBIC!";} print "\n";
                            close ITASSER;
                            open HYDROPHOBICITY, ">$itasserout/$_-$domains/hpstatus.txt" or
die$!;
                            print HYDROPHOBICITY "$hpstatus";
                            system ("$stridebin/stride $itasserout/$_-$domains/seq.fasta >
$itasserout/$_-$domains/stride.out");
                            $hpbcount=0;
                            $bound++; sleep(1);
                    }
            }
$dombound = 0;
}
```

6.3 Appendix D: Solvent Accessibility, Hydrophobicity, Nearest Neighbor Calculation

```
#!/usr/bin/perl
use warnings;
use strict;
use tempmodule;
use File::Path qw(mkpath);
my $neighbor;
my $numdom;
my $bound;
my $firstline;
my $fasta;
my $domseq;
my $dombound = 0;
my $hpbcount = 0;
my $domlength;
my $neighborthreshold = 3;
my $neighbornum;
my $hppthreshold = 1.6;
my $residue;
my $hpproduct;
my @neighborarr;
my $neighbors;
my $combonum = 1;
my @combopatch;
my %nonqual;
my %qual;
my %patches;
my @patcharr;
my $neighborradius = 8;
my %hydrophobic = (
        ILE => 4.5,
        VAL => 4.2,
        LEU => 3.8,
        PHE => 2.8,
        CYS => 2.5,
        MET => 1.9,
        ALA => 1.8,
        GLY => -0.4,
        THR => -0.7,
        TRP => -0.9,
        SER => -0.8,
        TYR => -1.3,
        PRO => -1.6,
        HIS => -3.2,
        GLU => -3.5,
        GLN => -3.5,
        ASP => -3.5,
        ASN => -3.5,
        LYS => -3.9,
        ARG => -4.5,
        X => 0.0
);



my %surface = (
        ILE => 182,
        VAL => 160,
        LEU => 180,
        PHE => 218,
        CYS => 140,
        MET => 204,
        ALA => 113,
        GLY => 85,
        THR => 146,
        TRP => 259,
        SER => 122,
        TYR => 229,
        PRO => 143,
        HIS => 194,
        GLU => 183,
        GLN => 189,
        ASP => 151,
        ASN => 158,
        LYS => 211,
        ARG => 241,
        X => 120
);

#read in domains
foreach(@proteinlist) {
            open INFILE,  "$outdir/$_/$_.result" or die "cannot open Threadom results";
            $firstline = <INFILE>;
            close INFILE;

            if ($firstline =~ /^\S+\s+\S+\s+(\d+)/) {$numdom = $1};
            # print $numdom; print "\n";
            $bound="1";

            open FASTA, "$outdir/$_/$_.fasta" or die$!;
            $fasta = <FASTA>;
            close FASTA;
        for (my $domains = 1; $domains <= $numdom; $domains++) {
                if ($firstline =~ /\($bound\-(\d+)\)/){
                        $bound = $1;
                        $dombound = $bound - $dombound;

                        open ITASSER, "$itasserout/cluster1/$_-$domains/model1.pdb" or die$!;
                system ("$stridebin/stride $itasserout/cluster1/$_-$domains/model1.pdb>
$itasserout/$_-$domains/stride.out");
                print "STRIDE output for $_-$domains generated\n";

                open STRIDE, "$itasserout/$_-$domains/stride.out";
                $hpbcount = 0; my $rescount = 0;
                my @areares; my @resarr;

                while (my $line = <STRIDE>) {

                     if ($line =~ /-\s*(\d+).+ (\d+.\d)      ~~~~/) {
                           push @resarr, $2; push @areares, $1;
                           }

                     }
                close STRIDE;
                        open CAMODEL, "+>$itasserout/$_-$domains/CAmodel.out";
                        my $lines; my @residuearr; my @resnumarr; my @xcoordarr; my
@ycoordarr; my @zcoordarr; my %exposed;
                        while ($lines = <ITASSER>) {
                                if ($lines =~
/^ATOM\s+\d+\s+CA\s+(\w\w\w)\s+(\d+)\s+(-?\d+.\d+)\s+(-?\d+.\d+)\s+(-?\d+.\d+)\
s+/) {
                                                my $residue = $1; push @residuearr,$residue;
                                                my $resnum = $2; push @resnumarr,$resnum;
                                                my $xcoord = $3; push @xcoordarr,$xcoord;
                                                my $ycoord = $4; push @ycoordarr,$ycoord;
                                                my $zcoord = $5; push @zcoordarr,$zcoord;
                                                print CAMODEL $lines;
                                        if ($resarr[$resnum-1]/$surface{$residue} >= 0.5)
{$exposed{$resnum}=" ";}
                                        }
                                }
                        close ITASSER;
                        open PATCH, ">$itasserout/$_-$domains/patch.out" or die$!;
                        foreach my $res1 (@resnumarr) {
                                printf PATCH "%-4d", "$res1"; print PATCH " Neighbors: ";
                                my $hpproduct =0; my $neighbornum = 0; my $hpneighbors = 0;
my $nearestneighbors = '';
                                foreach my $res2 (@resnumarr)  {
```

```
                                        if ((($xcoordarr[$res1-1]-$xcoordarr[$res2-
1])**2+($ycoordarr[$res1-1]-$ycoordarr[$res2-1])**2+($zcoordarr[$res1-1]-$zcoordarr[$res2-
1])**2)**(0.5) <= $neighborradius && (($xcoordarr[$res1-1]-$xcoordarr[$res2-
1])**2+($ycoordarr[$res1-1]-$ycoordarr[$res2-1])**2+($zcoordarr[$res1-1]-$zcoordarr[$res2-
1])**2)**(0.5) > 0 && exists($exposed{$res2})) {
                                #       print PATCH "$resnumarr[$res2-1],";
                                        $nearestneighbors = $nearestneighbors . $resnumarr[$res2-
1] . ',';

                                        $neighbornum++;
                                        if ($hydrophobic{$residuearr[$res2-1]} >= $hppthreshold
&& exists($exposed{$res2})) {$hpneighbors++;}
                                        $hpproduct = $hpproduct +
$hydrophobic{"$residuearr[$res2-1]"};
                                        }
                                }
                                printf PATCH "%-20s", $nearestneighbors;
                                print PATCH " Number of neighbors: $neighbornum",
"[$hpneighbors]";
                                print PATCH " Hydrophobic Product SUM: "; printf PATCH
"%5s", $hpproduct;
                                print PATCH " Hydrophobic Product IND: "; printf PATCH
"%4s" ,$hydrophobic{$residuearr[$res1-1]};
                                print PATCH " Hydrophobic Product AVG: "; printf PATCH "%
0.2f", $hpproduct/($neighbornum+1); print PATCH "\n";
                        }

                        close PATCH; open PATCH, "$itasserout/$_-$domains/patch.out" or die$!;

                        open COMBO, "+>$itasserout/$_-$domains/combopatch.out" or die$!;
#begin temp
                        my @patcharray = <PATCH>; my @all; my @allneighbor; my
@hpproducts; my @neighbornums;my @groupscore; my $groups;
                        foreach $lines (@patcharray) {
                                if ($lines =~ /^(\d+)\s*Neighbors: (\S*)\s*Number of neighbors:
(\d+).+\s*Hydrophobic Product SUM:\s*(-?\d+.*\d*) Hydrophobic Product
IND:\s*(-?\d+.\d)/) {
                                        $residue = $1;
                          $neighbornum = $3;
                          $hpproduct = $5;
                          $neighbors = $2;
                                        $groups = $4;
                          }
                                        push @all, $residue; push @allneighbor, $neighbors;
                                        push @hpproducts, $hpproduct; push @neighbornums,
$neighbornum;
                                        my @group = split(',',$neighbors);
                                        #return here
                        }
                        foreach (@all) {
                                my $hpneighborcount = 0; my $friend;
                                my @indneighbors = split(',',$allneighbor[$_-1]);
                                foreach $friend (@indneighbors) {
                                        if ($hpproducts[$friend-1]>=$hppthreshold)
{$hpneighborcount++;}
                        }
                                if ( exists($exposed{$_}) && $hpneighborcount >=
$neighborthreshold && $hpproducts[$_-1] >= $hppthreshold) {
                        $qual{$_}="";
                        }
                        }

                        my @patch; my @cascade; my $neighbor; my $combonum = 1; my $amino;
                        close CAMODEL;
                        open ITASSER, "$itasserout/$_-$domains/CAmodel.out" or die$!;
                        open BFACTOR, "+>$itasserout/$_-$domains/bfactors.out" or die$!;
                        my @bfactor = <ITASSER>; foreach (@bfactor) {print BFACTOR substr
$_,0,62; print BFACTOR "\n";}
                        close BFACTOR; open BFACTOR, "$itasserout/$_-$domains/bfactors.out"
or die$!;
                        @bfactor = <BFACTOR>; my %patchmembers;
                        foreach $amino (@all) {
                                if (exists ($qual{$amino})) {
                                push @patch, $amino; #print "residue", "$_";<>;
                                push @cascade, split(",",$allneighbor[$amino-1]); #foreach
(@cascade) {print "cascade", "$_ ", "\n";} <>;
                                }

                                foreach $neighbor (@cascade) {
                                        if (exists ($qual{$neighbor})) {
                                        push @patch, $neighbor;
                                        push @cascade, split(",",$allneighbor[$neighbor-1]);
                                        @cascade = &uniq2(@cascade); #foreach (@cascade) {print
"cascade", "$_ ";} <>;
                                        }
                                }

                        @patch = &uniq2(@patch); @patch = sort(@patch);
                        if (@patch && !exists($patches{@patch}) && scalar(@patch) >=3 ) {
                                print COMBO "ComboPatch$combonum-";
                                my $patcharea = 0; my $patchsize=0; my $totalhydrophobicity=0;
                                foreach (@patch) {
                                                print COMBO "$_,"; $patchmembers{$_}='';
                                                $patcharea = $patcharea + $resarr[$_-1];
                                                $patchsize++;
```

```
        $totalhydrophobicity=$totalhydrophobicity+$hydrophobic{$residuearr[$_-1]}
                                        }
                                print COMBO "\nRESIDUES: $patchsize", "\nTOTAL AREA:
$patcharea", "\nAVG HYDROPHOBICITY: ", $totalhydrophobicity/$patchsize;
                                print COMBO "\nSCORE: ",
$totalhydrophobicity*$patcharea/$patchsize, "\n\n"; $patcharea=0; $patchsize=0;
$totalhydrophobicity=0;
                                $patches{@patch} = '';
                                $combonum++;
                                }
                                undef @patch; undef @cascade;
                                        if (exists($patchmembers{$amino})) {
                                                chomp $bfactor[$amino-1];
                                                $bfactor[$amino-1] = $bfactor[$amino-1] . " " .
($hpproducts[$amino-1]+4.5)*16;
                                        } else {
                                                chomp $bfactor[$amino-1];
                                                $bfactor[$amino-1] = $bfactor[$amino-1] . " " .
"0.00";
                                        }
                        }
                        close BFACTOR; open BFACTOR, ">$itasserout/$_-
$domains/bfactormodel.pdb" or die$!;
                        foreach (@bfactor) {print BFACTOR $_, "\n";} close BFACTOR; my
@bfactors = @bfactor;
                        close ITASSER; open ITASSER, "$itasserout/cluster1/$_-
$domains/model1.pdb" or die$!;
                        open BFACTOR, "+>$itasserout/$_-$domains/bfactors.out" or die$!;
                @bfactor = <ITASSER>; foreach (@bfactor) {print BFACTOR substr $_,0,62;
print BFACTOR "\n";}
                close BFACTOR; open BFACTOR, "$itasserout/$_-$domains/bfactors.out" or
die$!;
                        @bfactor = <BFACTOR>;
                        foreach $amino (@all) {
                                if (exists($patchmembers{$amino})) {
                                                my $a; for ($a=0; $a< scalar @bfactor; $a++) {
                                                        if ($bfactor[$a] =~ / \w\w\w\s+(\d+) /) {
                                                                if ($amino==$1) {chomp
$bfactor[$a]; $bfactor[$a] = (substr $bfactor[$a],0,62) . " " . ($hpproducts[$amino-1]+4.5)*16;}
                                                        }
                                                }
                                        } else {
                                                my $b; for ($b=0; $b < scalar @bfactor; $b++){
                                        if ($bfactor[$b] =~ /1.00  $/) {chomp $bfactor[$b];
$bfactor[$b] = (substr $bfactor[$b],0,62) . " " . "0.00"}
                                                        }
```

```
                                                    }
                                    }
                            open FULL, ">$itasserout/$_-$domains/fullmodel.pdb" or die$!;
                            foreach (@bfactor) {print FULL $_; print FULL "\n";}
                            close COMBO; undef %qual; undef %exposed; undef %patches;
                            $hpbcount=0;
                            $bound++;
                    }
            }
$dombound = 0;
}

sub intCheck{

my $num = shift;

return ($num =~ m/^\d+$/);

}
sub round {

my $var = shift;

if (intCheck($var - 0.5)) { $var = $var + 0.1; }

return sprintf("%.0f", $var);

}

sub uniq2 {
   my %seen = ();
   my @r = ();
   foreach my $a (@_) {
      unless ($seen{$a}) {
          push @r, $a;
          $seen{$a} = 1;
      }
   }
   return @r;
}
```

Chapter 7

Acknowledgements

I would like to thank Dr. David Shultis for initially introducing me to the wonderful world of programming and bioinformatics research. The many, many weeks I have spent fixing problems in my code and running test cases only to discover new problems have definitely shaped my career goals and have been one of the most formative experiences in my undergraduate career. I would also like to thank David for the plethora of letters of recommendation which have landed me both summer employment and medical school admission, both for which I am very grateful. I would like to thank Professor Noemi Mirkin for agreeing to be my Biophysics co-sponsor for this research despite my repeatedly falling asleep in her class (I still blame morning rowing practices) as well as (hopefully!) ignoring that fact when writing my letters of recommendation. Finally, I would like to give special thanks to every member of UMRT with whom I have ever had the honor of sharing a boat or erg room. Without their continual inspiration, I would never have developed the dedication and discipline necessary to pursue any kind of research, let alone a career in it. While the end of my undergraduate rowing career may quickly be approaching, the pursuit of meters will continue. Who knows? Maybe one day they may actually show up... Godspeed.