Automation of PRISM Algorithm for Prediction of Protein-Protein Interactions

by

Pelin Atıcı

A Thesis Submitted to the

Graduate School of Sciences and Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Computer Sciences and Engineering

Koc University

August 2013

Koc University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Pelin Atıcı

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

____

Prof. Attila Gürsoy (Advisor)

____

Prof.Özlem Keskin

____

Assoc. Prof. Engin Erzin

Date: ____________________

ABSTRACT

Protein - protein interactions among numerous proteins are the building blocks of the biological networks. To understand the function of a cell, it is necessary to examine biological processes occuring in the cell. Many biological processes in the cell depend on these biological networks, thus researchers in the field of biology always give attention to these networks and interactions. There are many experimental methods available to predict protein-protein interactions. Also, many computational methods have been developed for this aim, especially in recent years.

In this project, we present a web server for the automation of a protein-protein interaction prediction algorithm. PRISM (Protein Interactions by Structural Matching) algorithm predicts interactions among various proteins by using structural and evolutionary similarity to known template interfaces. Our software system constructs a relational database of pre-calculated protein-protein interaction predictions, and the template and target structures used to calculate these predictions. It involves three sections which are templates, targets and predictions. Contents of the database can be queried using the appropriate PRISM web page. Also, visualization of these template and target structures are generated using Jmol plug-in in our web server.

Also, another important aim of our web server is to run the PRISM algorithm from scratch when the protein structures in hand are not present in the relational database. In the case of such a situation, the steps of the algorithm are executed in an orderly fashion and all manual operations between these steps are eliminated. Computation of PRISM algorithm can take a lot time and effort. Using a 4-tier web database application architecture and a message queue system, many computations can run concurrently and users do not have to do any manual operations during the computation aside from input their protein structures. Since the manual operations are eliminated, users do not have to wait one step to finish to continue to the next step of the algorithm.

ÖZET

Çok sayıda protein arasındaki etkileşimler biyolojik ağların yapıtaşlarını oluşturur. Bir hücrenin fonksiyonunu anlamak için, hücre içinde gerçekleşen biyolojik süreçleri incelemek gerekir. Hücredeki birçok biyolojik süreç de biyolojik ağlara bağlıdır. Bu yüzden biyoloji alanındaki araştırmacılar her zaman bu ağlara ve etkileşimlere önem vermiştir. Proteinler arasındaki etkileşimleri öngören birçok deneysel yöntem mevcuttur. Ayrıca özellikle son yıllarda, bu amaç için birçok hesaplamalı yöntem de geliştirilmiştir.

Bu projede, protein etkileşimlerini tahmin etmek için kullanılan bir algoritma bir internet sunucusu aracılığıyla otomatik hale getirilmiştir. PRISM algoritması, bilinen şablon protein arayüzlerine yapısal ve evrimsel benzerlik kullanarak çeşitli proteinler arasındaki etkileşimleri tahmin eder. Bizim yazılımsal sistemimiz, önceden hesaplanmış protein etkileşimlerinden ve bu etkileşimleri hesaplamak için kullanılan şablon ve hedef yapılarından oluşan ilişkisel bir veritabanı oluşturur. Yazılımsal sistem; şablonlar, hedefler ve tahminler olmak üzere üç bölümden oluşur. Veritabanının içeriği uygun PRISM internet sayfası kullanılarak sorgulanabilir. Ayrıca, JMol eklentisi kullanılarak şablon ve hedef yapıların görselleştirilmesi sağlanmıştır.

İnternet sunucumuzun bir başka önemli amacı, mevcut protein yapıları ilişkisel veritabanımızda bulunmadığı zaman PRISM algoritmasını en baştan çalıştırmaktır. Bu durumda, algoritmanın adımları sırasıyla uygulanır ve bu adımlar arasındaki elle yapılan bütün işlemler elenmiş olur. PRISM algoritmasının hesaplaması fazla zaman alabilir ve çaba gerektirebilir. Dört katmanlı bir internet veritabanı uygulama mimarisi ve mesaj sıra sistemi kullanılarak birçok hesaplama aynı anda çalışabilir ve kullanıcılar protein yapılarını girmek harici elle yapılan herhangi bir işlem yapmak zorunda kalmazlar. Elle yapılan işlemler elendiği için, kullanıcılar algoritmanın bir sonraki aşamasına geçmek için önceki aşamanın bitmesini beklemek zorunda kalmazlar.

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Dr. Attila Gürsoy and Prof. Dr. Özlem Keskin for their comments and support on my thesis. I would also like to thank Assoc. Prof. Engin Erzin for agreeing to be in my thesis jury and for his valuable feedback.

I would like to thank all my friends in Koc University; especially my housemate Aslı Pınar Yapıcı, my source of entertainment in the office, Büşra Topal, Tuğçe Yıldızoğlu and Engin Çukuroğlu, and my other office mates Serap Beldar, Emine Güven Mayorov, Selin Karagülle, Güray Kuzu who have always been eager to help and share their knowledge during the last three years.

Lastly, I would like to express my gratitude to my dear sister, my mother and my father who have always supported me during all my education.

TABLE OF CONTENTS

LIST OF TABLES .... **viii**

LIST OF FIGURES .... **ix**

NOMENCLATURE .... **x**

Chapter 1: INTRODUCTION **1**

Chapter 2: RELATED WORK **3**

2.1 Protein Interfaces 3

2.2 Protein-Protein Interaction Prediction 4

2.3 Web Servers and Databases 5

2.3.1 Web Servers for Prediction of Protein-Protein Interactions 5

2.3.2 Web Servers that are Similar from Usage and Software Architecture 7

2.3.3 The Old PRISM Web Server 12

2.4 Web-Database Application Architecture (3-tier) 13

Presentation Layer: 14

Business Layer: 14

Data Layer: 15

2.5 Interprocess Message Queues 15

2.6 Contributions of the Thesis 17

Chapter 3: METHODS 19

3.1 PRISM Algorithm 19

3.1.1 Template Dataset 19

3.1.2 Target Dataset 20

3.1.3 The Algorithm 21

3.1.4 PRISM Protocol 23

3.2 PRISM’s Web-Database Application Architecture (4-tier) 24

Chapter 4: RESULTS 27

4.1 Design of PRISM 27

4.1.1 Database Design 28

4.1.2 Application Architecture Design 33

4.1.3 Web Interface Design 36

Chapter 5: CONCLUSION 47

BIBLIOGRAPHY 49

LIST OF TABLES

Table 4.1 Description of ‘templates’ table 30
Table 4.2 Description of ‘targets’ table 30
Table 4.3 Description of ‘predictions’ table 31
Table 4.4 Description of ‘jobs’ table 32
Table 4.5 Description of ‘onlineRun’ table 33
Table 4.6 Description of ‘simpleCount’ table 33

LIST OF FIGURES

Figure 2.1 Example of a protein interface [11]. The residues which are shown with surface representation instead of ribbon representation indicate the interacting residues (interface area)........ 4
Figure 2.2 Three tier architecture........ 14
Figure 2.3 Simple Point-to-Point Messaging........ 16
Figure 2.4 Simple operation of RabbitMQ. “P” represents “Producer”, “C” represents “Consumer”. Messages published by producers are sent to a queue. Messages are stored on the queue. Consumers wait to receive messages........ 17
Figure 3.1 Construction of template dataset [34]........ 20
Figure 3.2 Construction of target dataset [34]........ 21
Figure 3.3 The algorithm of prism [37]........ 22
Figure 3.4 Web-Database Application Architecture of PRISM web server. Note that Business Layer is separated into two parts as Application Logic and Compute Server. 25
Figure 4.1 ER Diagram of PRISM Database........ 29
Figure 4.2 Application Architecture Design of PRISM Web Server........ 34
Figure 4.3 Templates Page........ 37
Figure 4.4 Jmol Visualization of a Template (1c4z)........ 38
Figure 4.5 Targets Page........ 39
Figure 4.6 Visualization of a Target (1i7k)........ 40
Figure 4.7 First two sections of the Predictions page........ 42
Figure 4.8 Page of all Predictions present in PRISM........ 43
Figure 4.9 Last three sections of the Predictions page........ 44
Figure 4.10 Results Page........ 45
Figure 4.11 Partners of Template Interfaces page........ 46

NOMENCLATURE

ASA	Accesible Surface Area
GUI	Graphical User Interface
HTML	HyperText Markup Language
PDB	Protein Data Bank
PHP	HyperText Preprocessor
PRISM	Protein Interactions by Structural Matching
WWW	World Wide Web

Chapter 1

INTRODUCTION

Proteins rarely act in isolation, and the complexity of organisms arise mostly from the interactions between the proteins (genes) of an organism rather than the number of proteins in it [1, 2]. Protein-protein interactions occur when two or more proteins bind together to carry out their biological function.

However, determining such a large scale problem is very hard both from the experimental view and from the computational view. Computational methods can guess the protein-protein interactions at different levels: they can predict the binding sites of proteins, find specific residues contributing the interaction, and design specific interfaces. Further, computational 'docking' [3-5] methods can predict protein-protein interactions. However, even without the computational costs, if there is a lack of biochemical information about the interaction sites, it is very difficult to predict the native interaction because there are many energetically-favorable ways for proteins to interact. Docking becomes much more challenging and computationally demanding if we want to apply it on the whole genome when we do not know which proteins interact. An alternative strategy is to apply structural similarity to an interface of a known protein complex.

In light of the information that the interface regions of proteins are more conserved than their overall structures [6], and also protein pairs with different structures and functions can associate via similar interface architectures [7-9], using interface structures can produce promising models for protein complexes even in the absence of global sequence or fold similarity. PRISM (Protein Interactions by Structural Matching) is an algorithm for finding interactions between the proteins based on this idea. In this work, we design a web server to predict interactions between input target proteins using PRISM algorithm.

Related work about the subject, which includes definition of protein interfaces and methods to predict protein-protein interactions, is presented in Chapter 2. In this section, web servers about proteins which are similar to the PRISM web server on content and design is explained too. Lastly, typical web-database applications and interprocess message queue systems are reviewed.

In Chapter 3, we present the details of the PRISM algorithm and web-database application architecture used in the design of the web server. Firstly, we describe the construction of datasets that are used in the algorithm. Then PRISM protocol, a downloadable protocol using PRISM algorithm to predict interactions between proteins, is explained in terms of usage. With this, we aim to show the steps that are eliminated when using the web server. Lastly, the web-database application architecture used in the design of PRISM web server and its difference from other typical architectures is explained.

The details about the implementation of web server and the result pages are provided in Chapter 4. The entity relationship diagram of the PRISM database and description of the tables is presented. Then, different sections of PRISM web server is described with screenshots and instructions. Example pages of visualization, query and online run are provided.

In the final chapter, a brief summary of the work done in this thesis is given.

Chapter 2

RELATED WORK

This chapter includes information about the subjects which are necessary to understand the studies in this thesis. These subjects are about protein interactions and web servers that are similar in content to the web server implemented in the scope of this thesis.

2.1 Protein Interfaces

Proteins rarely function seperately. Instead, they generally interact to fulfill their biological functions and form protein complexes. The portion that takes part on the interaction of the protein is called the interface.

An interface is defined as interacting residues and nearby residues, respectively. If the distance between any two atoms belonging to two residues from different chains is less than the sum of their van der Waals radii plus 0.5 A, these two residues are defined as 'interacting'; if the distance between the Cα of a non-interacting residue and an interacting residue in the same chain is under 6 A, the noninteracting residue is flagged as a 'nearby' residue [10]. An example of an interface can be seen in Figure 2.1.

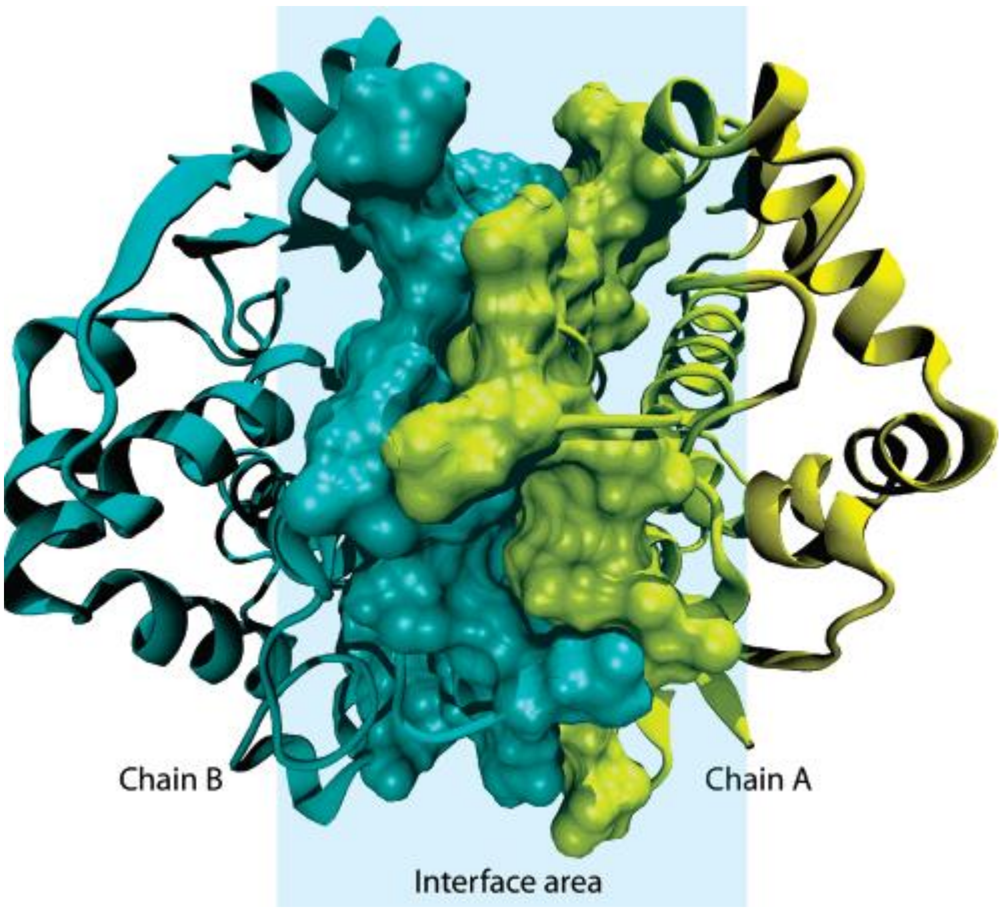


Figure 2.1 Example of a protein interface [11]. The residues which are shown with surface representation instead of ribbon representation indicate the interacting residues (interface area).

2.2 Protein-Protein Interaction Prediction

The interactions among proteins rather than the number of proteins in an organism indicate the complexity of it. Hence, understanding the interaction between pairs or groups of proteins is important to predict the function of proteins. There are different experimental methods to predict protein-protein interactions such as yeast two-hybrid systems, affinity purification/mass spectrometry, protein-fragment complementation assays (PCA), protein and DNA microarrays, fluorescence resonance energy transfer (FRET) and Microscale Thermophoresis (MST).

However, experimental methods sometimes lead to false positives or false negatives. They are biased towards certain protein types and cellular localizations. Also, they tend to detect higher affinity interactions since they are more stable and last for long periods of time whereas lower affinity interactions behave in the opposite direction. As might be expected, these methods also have a high cost in both time and expense. The limitations of the experimental methods create the need of the computational methods for predicting protein-protein interactions.

In recent years, computational methods are also developed to predict protein-protein interactions. These methods can guess the protein-protein interactions at different levels: they can predict the binding sites of proteins, find specific residues contributing the interaction, and design specific interfaces. Further, computational 'docking' methods can predict protein-protein interactions. A great number of computational tools have been developed based on the assumption that proteins which are functionally related most probably interact physically. Computational approaches can be classified in five categories: genomic methods, evolutionary relationship, protein structure based methods, domain-based methods, protein sequence based methods.

2.3 Web Servers and Databases

This section is about the similar web servers to the one implemented in the scope of this thesis in terms of content and implementation. This list includes web servers which predict interactions between proteins with different methods, implement docking between protein structures, predict specific residues of proteins(e.g. ligand-binding, catalytic, interface residues) or visualize gene networks. They generally use PDB formatted files for input proteins and implement a job queue system to run jobs in order.

2.3.1 Web Servers for Prediction of Protein-Protein Interactions

PredUS

PredUS is a web server that predicts protein-protein interfaces of potential locations at which proteins interact with other proteins [12]. It uses template-based prediction. Given a query protein, an interface of its inferred based on some similarity to another protein. (Given a query protein, PredUS ‘map’ interaction sites of structural neighbors involved in a complex to residues on the surface of the query.)

PredUS server’s algorithm is composed of two steps. In the first step, structural neighbours are found using a structural alignment program. After that, contact frequencies of all residues of the query protein are compared with structural neighbours.

Users can enter input protein structure as a PDB formatted file or a PDB code. After PredUS validates the input structure, users can either wait or provide an e-mail adress and job id to retrieve the results. As an output, list of residues and their scores downloadable in text format are generated. Users also visualize individual predictions in which resudies are colored according to the residue score.

PredUS is accessible from http://bhapp.c2b2.columbia.edu/PredUs/.

PRED_PPI

PRED_PPI is a web server for predicting protein-protein interactions with probability assignments [13]. It uses a sequence based approach to predict interactions among proteins.

Users enter two protein sequences in FASTA format, chooses an organism-specific predictor(database) for the query proteins. Then, a probability threshold is chosen between 0 and 1 whose default value is 0.5.

The result page reports whether the entered proteins interact or not under the chosen probability threshold. Also, users are provided with the actual probability of the interaction between their query proteins.

PRED_PPI is available at http://cic.scu.edu.cn/bioinformatics/predict_ppi/default.html.

Struct2Net

Struct2Net is a web server for predicting protein-protein interactions using a structure-based approach [14]. It combines a threading approach for template alignment with a machine learning approach to estimate a score for the interaction.

Users can directly retrieve the pre-computed predictions for the most commonly studied organisms (saccharomyces cerevisiae, drosophila melanogaster, homo sapiens) by entering gene name/id or a keyword. For the other organisms, users can query by entering protein sequences in FASTA format. Users also can upload a file containing multiple sequences in FASTA format.

Users have the option to get the results fast but approximately using orthology over pre-computed interactions and slower but with full computation. This option is available in the user interface page. When querying the pre-computed interactions, results are returned instantaneously. If users select fast-but-approximate option, completion of the submitted job only lasts till twenty seconds. Slower but full computation lasts for fourty five mins. Users can enter their e-mail adresses for the jobs with full computation to retrieve the link of the results page when the run is finished. Also, users can check the ongoing job's progress using job id from the 'Fetch job' section.

In the results page, for the querying pre-computed interactions, each prediction with its confidence score is listed. Also, there are links for each gene to view their GO annotations.
For predictions by threading sequences onto all templates, the output page shows the greatest probability for the interaction between the query proteins. For a potential interaction, template-sequence alignments for best-fit complex templates are shown along with their confidence scores and threading details. Confidence scores are between 0 and 1, where 0 indicates minimum confidence and 1 indicates maximum confidence.

Struct2Net is available at http://groups.csail.mit.edu/cb/struct2net/webserver/.

2.3.2 Web Servers with Similar Usage and Software Architectures

Kinari-Web

Kinari-Web is a web server for performing rigidity and flexibility analysis of proteins [15]. It predicts which groups of atoms (rigids clusters) are likely to move together using only inter-atomic connectivity and interaction information as different from molecular dynamics. Kinari-Web is also used for visually exploring rigidity of proteins by using a Jmol-based 3D visualizer. Kinari-Web uses PDB formatted file for the calculations which contains protein structure data. Users can either upload a PDB formatted file or they enter a PDB code and Kinari-Web retrieves PDB file from the Protein Data Bank.

In the running phase, Kinari-Web analyzes the protein given and show some results about the protein to the users. Users can change some variables or modeling options and choose to recalculate or remodel interactions.

Users can download list of the rigidity analysis results (# of clusters, size of largest cluster etc.) after running phase or they can visualize the rigid clusters with different options.

Kinari-Web is accesible from http://kinari.cs.umass.edu.

PINTA

PINTA is a web server which identifies candidate genes related to a disease (gene prioritization problem) [16]. It uses the assumption that strong candidate genes tend to be surrounded by many differentially expressed genes and performs candidate gene prioritization starting from a pre-defined set of candidate genes.

PINTA web server completes its function in four steps. In the first step, the organism of interest is chosen among five organisms (human, mouse, rat, worm, yeast) that are available. In the second step, users choose the type of ranking whether it is genome-wide ranking or ranking the list of candidate genes. In the third step, users decide how to compare gene ranks. The last step involves choosing to use exemplary data or uploading expression data specific to a disease. After these steps, the method can be run with default setting or advanced setting. In advanced setting section, users can choose the ranking strategy and network type. As a result, PINTA generates a candidate gene ranking table that is downloadable.

PINTA is accesible from http://www.esat.kuleuven.be/pinta/.

FlexPepDock

FlexPepDock is a web server which uses a high resolution peptide docking protocol for the modeling of peptide-protein complexes [17]. It is important since peptide-protein interactions are common among the interactions in the cell.

The input for the web server is a PDB file of a complex between a protein receptor and peptide. After submitting the input file, users can either wait for the results or enter an e-mail adress to be notified when the results are ready.

The server will dock the peptide starting from the initial conformation in the submitted file. It will then rank the total 200 created models and provide users with the top 10 results with their scores and a plot of the energy landscape sampled by these 200 models. Another important feature of the web server is that it allows the users to see the status of their submitted jobs (queued, docking, failed, completed etc.). Also, it shows how many models have been created instantaneously in the job queue since the start of the submitted job.

FlexPepDock is accesible from http://flexpepdock.furmanlab.cs.huji.ac.il/.

Firestar

Firestar is a web server for predicting catalytic and ligand-binding residues in protein sequences [18, 19]. It predicts functional residues from the information extracted from remotely related structures. Alignments between query sequences and FireDB [20] templates are used to predict functional information using HHsearch [21] and PSI-BLAST [22] alignment search tools.

Users can enter a PDB code, a PDB coordinates file or a FASTA sequence as input. Also, they can change the e-value for the PSI-BLAST tool on the input screen. After submitting the input structure, users can wait for the results or enter an e-mail adress to retrieve the results later. The alignments generated are evaluated using SQUARE [23] to predict the functionally important resudies in the input protein structure.

As an output, a text summary which includes information for each predicted catalytic and bindig site, a static image of local alignment results (if input is a structure, structural alignment results) and a visualization of catalytic and ligand-binding residues are shown.

Firestar is available at http://firedb.bioinfo.cnio.es/Php/FireStar.php.

PRUNE and PROBE

PRUNE and PROBE are two web services for protein-protein docking [24]. Docking is also important for understanding the protein-protein interactions. Protein-protein docking algorithms generally includes four major tasks:

1. Generating of docking posses
2. Select a subset of posses
3. Structural refinement and scoring of selected posses
4. Ranking for the final assesment

PRUNE web server is used for the first and second tasks, whereas PROBE web server is used for the third and the forth tasks.

PRUNE:

PRUNE web server is used to select a subset of docking poses generated during sampling search using an edge-scoring function.

Users input a receptor coordinate file (PDB format), a ligand coordinate file (PDB format) and a file containing transformation matrix. Then, they choose the format of the transformation matrix (FTDock,Z-DOCK PatchDock, GrammX). After entering the inputs, users can wait or enter their e-mail adresses to retrieve the link of the result page later. The result page outputs transformation matrices of the generated poses in the same format as the input. Users can download the pruned poses and use a program to rank them. Also, there is an option to forward these generated poses to the PROBE server to be scored and ranked by it.

PRUNE is available at http://pallab.serc.iisc.ernet.in/prune/.

PROBE:

PROBE web server is used to refine, score and rank selected docking posses. As input, users may enter subset of poses generated by some docking technique or enter two unbound protein molecules to dock. As in the PRUNE server, users can wait fort he results or enter their e-mail adresses to retrieve the results later.

The results page includes a table of rank-sorted list of complexes with details on individual parameters and the final PROBE score values. Each predicted complex can be visualized using Jmol[25] software.

PROBE is available http://pallab.serc.iisc.ernet.in/probe/.

CSpritz

CSpritz is a web server for the prediction of protein disorder [26]. Disordered regions are non-folding or partially folding regions in proteins and they also have a functional importance.

As input, users enter single or multiple sequences in FASTA format by pasting the sequences as text or uploading files. After that, they select the prediction type which can be for short(x-ray) or long(disprot) disorder. In the running phase of the server, users can wait for the job to finish or they can choose to be notified when the results are ready by giving their e-mail adresses.

After users submit the job, a page that gives information about the current status of the job opens and it is refreshed in every thirty seconds. When the job finished running, the results are returned to the users on this page. These results include:

- Disorder plot (Residue index vs. Probability of disorder) as a PDF file
- Graph of homologues found (Residue vs. # of residues found in structural templates) as a PDF file
- Disorder prediction (Disorder probabilities) as a text file
- Protein statistics (Total amino acids, total # of disordered residues, # of disordered segments etc.)
- Statistics and graphs of each disordered segment as pdf and text files

CSpritz is accesible from http://protein.bio.unipd.it/cspritz/.

GeneMANIA

GeneMANIA is a web interface which generates hypotheses about gene function, analyzes gene lists and prioritizes genes for functional arrays (Extend query genes with functionally similar genes) [27]. It is mostly about visualization of gene networks.

The input to GeneMANIA is a list of genes. As advanced options, they can select the desired network, network weighting method and the number of genes to return. The results are loaded in a short time.

GeneMANIA extends the input gene list with the genes that are functionally similar and visualize an interactive functional association network, illustrating the relationships between genes (some relationships can be eliminated). On a side panel, networks of different types, genes and functions in the displayed network are listed and categorized. Users also can save the results of their analysis by the drop-down menu above the network. The network can be saved as text or vector image and genes, functions, interactions, attributes, search parameters can be saved, likewise.

GeneMANIA is available at http://www.genemania.org/.

2.3.3 The Old PRISM Web Server

The old PRISM Web Server is a website which is used to analyze protein-protein interfaces and protein-protein interactions [28]. It consists a database of protein interface structures derived from the Protein Data Bank (PDB) and a list of similarity matchings.

Unlike other web sites which are explained in the previous sections, PRISM is mostly a database of known protein-protein interactions. Users can query the contents of PRISM's database using different sections of the web server. The main way of accessing the contents of the database is by querying for a particular interface, target protein, or similarity matching, and then retrieve the specific details of these structures [29].

The main pages of the web server can be categorized into three types: 1) search pages, 2) result pages, and 3) details pages. Search pages contain a query form for users to search different entities. According to the entity type, the parameters of the search form differ. Result pages contains the results of the users' query. Contents of the results pages also differ according to the entity which is queried. After users select an entity among the results, details of this chosen entity are shown in details page. The details shown are also different for every type of entity chosen as expected.

In the aspect of usage, old PRISM web server contains three sections which are interfaces, targets and predictions. On the interfaces page, users can browse through clusters or template interfaces. Also, they can query the interfaces dataset using the search form. On targets page, users can list all targets present in the database or query a specific target protein by using the search form. Under the predictions section of PRISM web server, users can browse all predictions present in the database or search a prediction for specific proteins.

The old PRISM web server can also be used to predict new interactions between proteins. But this feature is limited to only one protein. In the Predictions section, Online Calculation part users enter a four character PDB id or the PDB structure file of their input protein. Then the web server begins to run PRISM protocol (see Section 3.1.4) for this target protein. After that, users should wait for the results without moving to any other section of the web server. If they close the current page where the calculation runs, then the results are inaccesible. Also different users can not access the web server during any calculation process.

2.4 Web-Database Application Architecture (3-tier)

In software engineering, web-database applications use different tiers to logically separate the presentation, business (application processing) and data management functions into different layers. This type of architecture are referred to as multi-tier architecture (n-tier architecture).

The most commonly used type of multi-tier architectures is three-tier architecture. Visual overview of this architecture can be seen in Figure 2.2.

Three-tier architecture is a client-server architecture in which the user interface, functional process logic (business rules), data storage and access are seperate modules [30]. In this kind of architecture, each tier should be independent from the others and should not include any dependency to other tiers at implementation. It means that these three tiers should not communicate.

The main advantage of using this type of architecture is that it allows any of the three tiers to be upgraded or altered according to the changing requirements of the application. It is also faster to develop such a system since work needs to be done can be distributed among the people with different duties (web designer for presentation, developer for business logic, db admin for data model).

Three-tier architecture is also the most secure architecture since the front-end users can not access the data because of the middle layer. By separating the business logic from the client, the client is only handles the presentation logic. Thus, only little communication is needed between the presentation and middle layer which makes front-end users to see and provide information fast and with no delay. An example of such a “thin” client is an Internet browser [31].

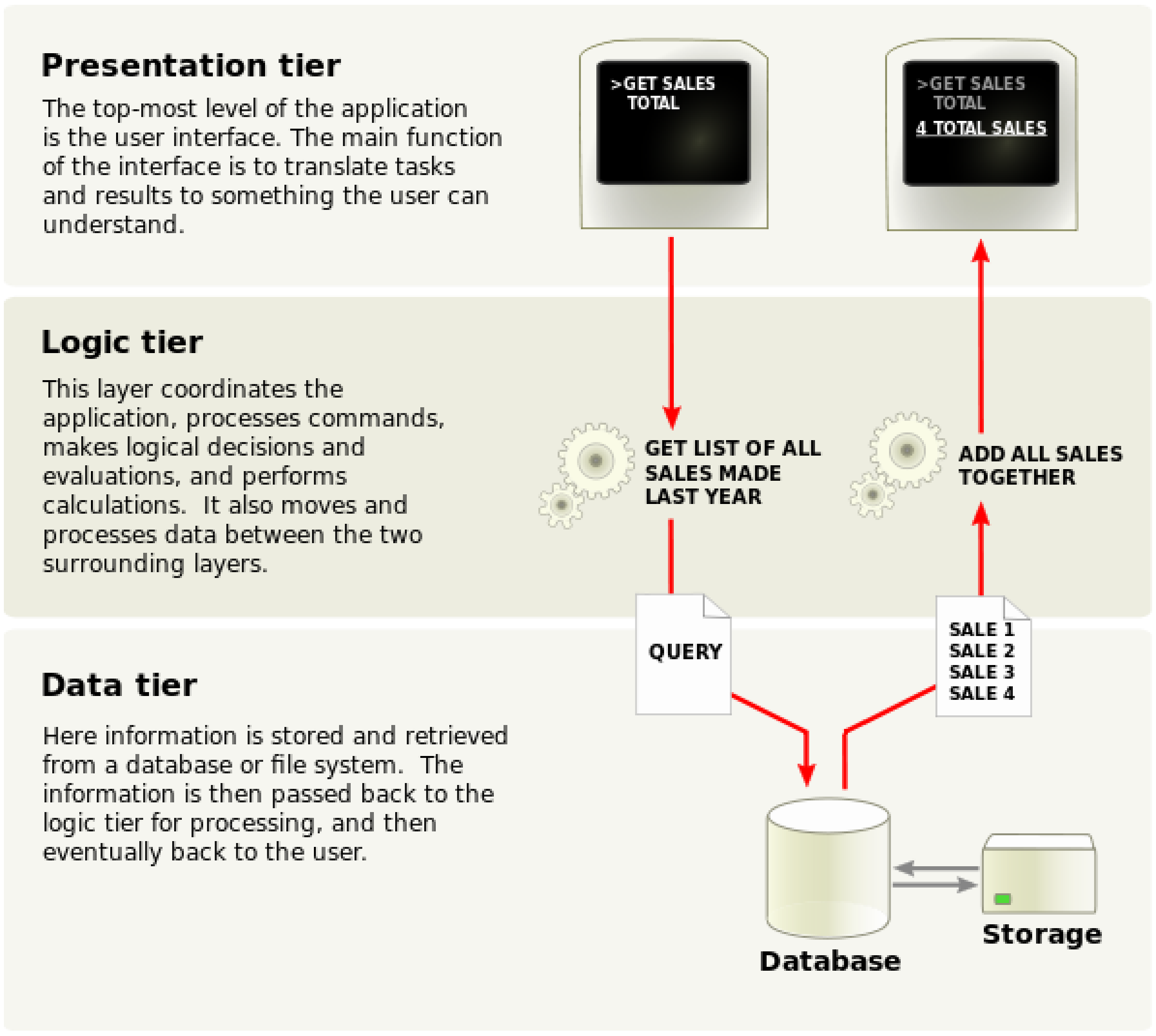

Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Database
Storage

Figure 2.2 Three tier architecture

Presentation Layer: It is the top most level of the three-tier application architecture which is often referred as GUI (graphical user interface), client view or front end. All interactions with the front users are handled in this layer. It communicates with the business tier and output results to the users through the browser/client architecture. This layer should not include any code about business logic or data access.

Business Layer: It is the middle layer of the three-tier application architecture which is often referred as middleware or back-end. This layer includes some set of rules for processing information. It communicates with both presentation layer and data access layer. It determines the logic to make the data in the data layer meaningfully available for the presentation layer.

In a web application this layer is a dynamic content processing and generation level application server (e.g. Java EE, ASP.NET, PHP). This layer should not include any code about presentation or data access.

Data Layer: It is the bottom level of the three-tier application architecture which is sometimes called as back-end. It handles the storage and retrieval of data from a database or file system. In a web application this layer is both a database and a database management system. This layer should not include any code about presentation or business logic.

2.5 Interprocess Message Queues

To create a scalable web server it is important to process asynchronous jobs. Using a message queue implementation, it is possible to put the jobs in order and process them when their time comes. Message queue systems use a queue for messaging (e.g. delivering content between processes) and provide asynchronous inter process communication. In this type of communication, one process acts as a client and sends a message to a specific queue. This queue stores messages until they are ready to be retrieved by the recipient. Since it is an asynchronous communication, the two processes (sender and receiver of the message) do not need to interact with the message queue at the same time. The other process acts as a server, retrieves the message from the specified queue and handles it. The message queue system takes care of everything in between. A simple overview of such a system can be seen in Figure 2.3.

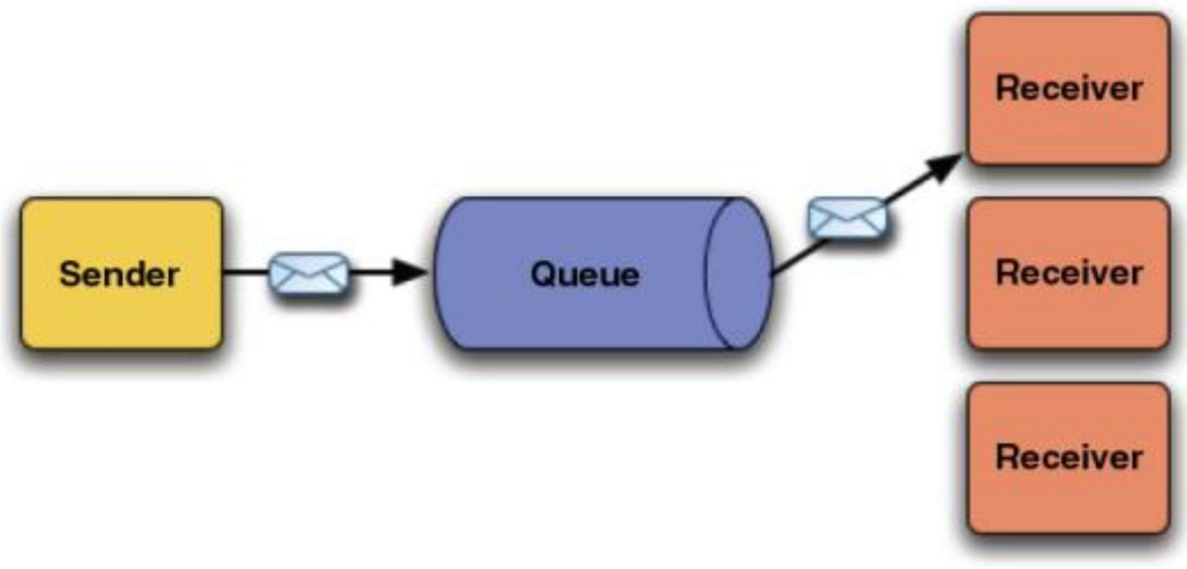


Figure 2.3 Simple Point-to-Point Messaging

Message-oriented middleware is a kind of implementation of message queue systems. It is a software or hardware platform for sending and receiving messages between systems. These implementations are also called as message brokers. The middleware creates a distributed communications layer and separates application developers from the details and complexity of master-slave nature of client/server mechanism. The other advantage of message-oriented middleware systems is that they support asynchronous communication. With this feature, if the receiver fails for any reason the sender can continue sending messages without affected of this situation. The messages sent will be collected in the message queue for later processing. The main disadvantage of these systems is that they require an extra component in the architecture which is the message broker.

There are a lot of open source message-oriented middleware systems in the market. Some examples of these are OpenAMQ, Apache QPid, Apache ActiveMQ, JORAM and RabbitMQ. RabbitMQ [32] came to the fore with its scalability, stability and simplicity among them. It is a robust messaging system (message broker) based on the AMQP (Advanced Message Queuing Protocol) [33]. AMQP is a binary protocol and defined set of rules to transmit application messages between two systems.

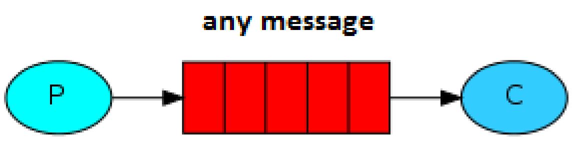


Figure 2.4 Simple operation of RabbitMQ. “P” represents “Producer”, “C” represents “Consumer”. Messages published by producers are sent to a queue. Messages are stored on the queue. Consumers wait to receive messages.

The basic working principle of RabbitMQ can be seen in Figure 2.4. Producer is a program that sends messages and consumer is a program that mostly waits for messages. A queue is like a message box and it lives inside RabbitMQ. It has no limit for the storage of messages. In a typical message-queueing implementation, a developer defines a named queue after configuring message broker. After that, the producer (an application that publishes messages) send message to this queue and RabbitMQ stores the messages until a receiving application connects. Then, the consumer (an application that processes messages) receives messages from that queue and processes them properly.

2.6 Contributions of the Thesis

Within this work, we implemented a web server which is mainly used to predict protein-protein interactions. To carry out this function, we automated an existing prediction algorithm and a protocol which uses this algorithm for ease of use and saving time. We believe that this research offers the scientific community to understand interactions between proteins with a useful tool.

Recapitulating, this work has made the following contributions:

- We eliminated all manual operations to run the PRISM protocol (see Section 3.1.4) which uses the PRISM algorithm for the calculation of interactions. These manual operations can be changing directory, giving parameters by hand when running some python scripts, copying some result files to another folder to be able to continue to the next step, replacing some files with the files prepared by hand.

- We removed the necessity of installing any external program before beginning to use the PRISM protocol. By doing this, we eliminated the possibility of version mismatches of compilers and users' making mistakes during the installation process. By this, we prevented the disfunction of external programs.

- We implemented a message queue system, aiming for to separate the database operations and the computation process from the application logic. This is done because computation of the algorithm can take a long time according to the number of target proteins and the template set used. Since calculation takes a lot of time, there is a bottleneck in the middle layer of the architecture. To eliminate this bottleneck, computation part is separated from the application logic and users are saved from the necessity of waiting.

- We have provided concurrent access of more than one users to the web server at the same time and during any calculation process. As mentioned, calculation can take a long time according to the number of input proteins. Being forced to wait for a calculation to finish can cause a lot of difficulty to the users when using the web server. By implementing a job queue system, we eliminated such a situation.

Chapter 3

METHODS

Prediction of protein-protein interactions between two or more protein structures is a current and important problem as discussed in Section 2.2.

In this work, we find the interactions between two proteins in a way that is fast and easy to use. To accomplish this goal, we designed a web server which uses PRISM algorithm (details are explained in Section 3.1) to predict protein-protein interactions. It is a standart web-database application which uses a job queue system. The algorithm, the job queue system and the general architecture used in the implementation of PRISM web server will be presented next.

3.1 PRISM Algorithm

PRISM (Protein Interactions by Structural Matching) [10, 28] is a system which employs a novel prediction algorithm for protein-protein interactions. It predicts interactions and binding residues between target proteins by using a structural and evolutionary similarity to known template interfaces. The method consists of two components: rigid body structural comparisons of target proteins to a set of protein-protein interfaces and flexible refinement and scoring using a docking energy function [34].

PRISM uses two types of datasets: a template dataset and a target dataset. The first one is a subset of a nonredundant dataset of known protein-protein interfaces derived from the PDB; the second one is a set of the structures of protein chains in a target cellular pathway.

3.1.1 Template Dataset

As can be seen in Figure 3.1, for the construction of the template dataset, by using the description for an interface, all interfaces between two protein chains obtained from protein assemblies of all types available in the PDB database were extracted. As a result, 49,512 two-chain interfaces are obtained and grouped into 8,205 clusters by their architectural similarity. Each cluster includes a representative interface structure and members similar to the representative interface [35].

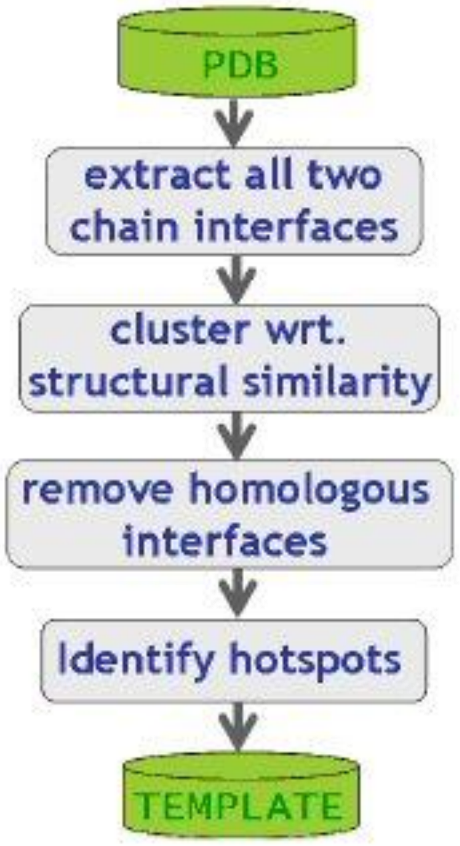

PDB
extract all two chain interfaces
cluster wrt. structural similarity
remove homologous interfaces
Identify hotspots
TEMPLATE

Figure 3.1 Construction of template dataset [35]

3.1.2 Target Dataset

Figure 3.2 shows the work flow of the construction of the target dataset. Firstly, all the polypeptide chains and complexes existing in the PDB were extracted. Every pair of member structures in this dataset is used for testing for potential interactions. A list of 10,158 proteins is obtained by downloading the set of proteins obtained by applying a sequence identity filter of 50% to all existing protein structures in the PDB. Then, the multimeric proteins are split into constituent chains and the target dataset consists of 18,698 structures [35].

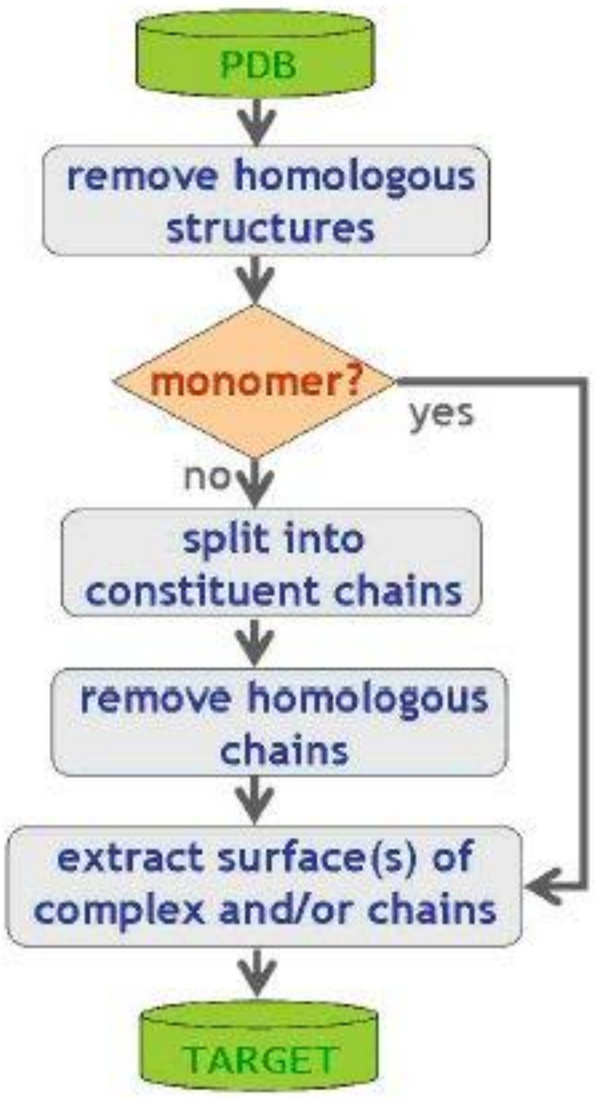

PDB
remove homologous structures
monomer?
yes
no
split into constituent chains
remove homologous chains
extract surface(s) of complex and/or chains
TARGET

Figure 3.2 Construction of target dataset [35]

3.1.3 The Algorithm

Since protein pairs with different global structures and different functions can associate via similar interface architectures [8], using interface structures can produce promising models for protein complexes even in the absence of global sequence or fold similarity.

The algorithm of PRISM arises from this concept: ***if the two complementary sides of a template interface are similar to the surfaces of two target proteins, then these two proteins can interact with each other using this template interface architecture.*** Visual explanation and work flow of the algorithm can be seen in Figure 3.3.

The algorithm consists of four phases:

1. The surface regions of target proteins are extracted.

2. The similarity of each side of a known interface to monomer surface regions is evaluated by using structural alignments (MultiProt) [36, 37].

3. The two chains whose surface regions are similar to the two parts of the template interface are transformed onto this template forming a complex structure, and the solution is assessed.

4. Flexible refinement of the rigid docking solutions of MultiProt is done to resolve steric clashes, and to rank the putative complexes by the global energy by FiberDock [34]. The predicted protein complexes are ranked according to these results obtained from FiberDock.

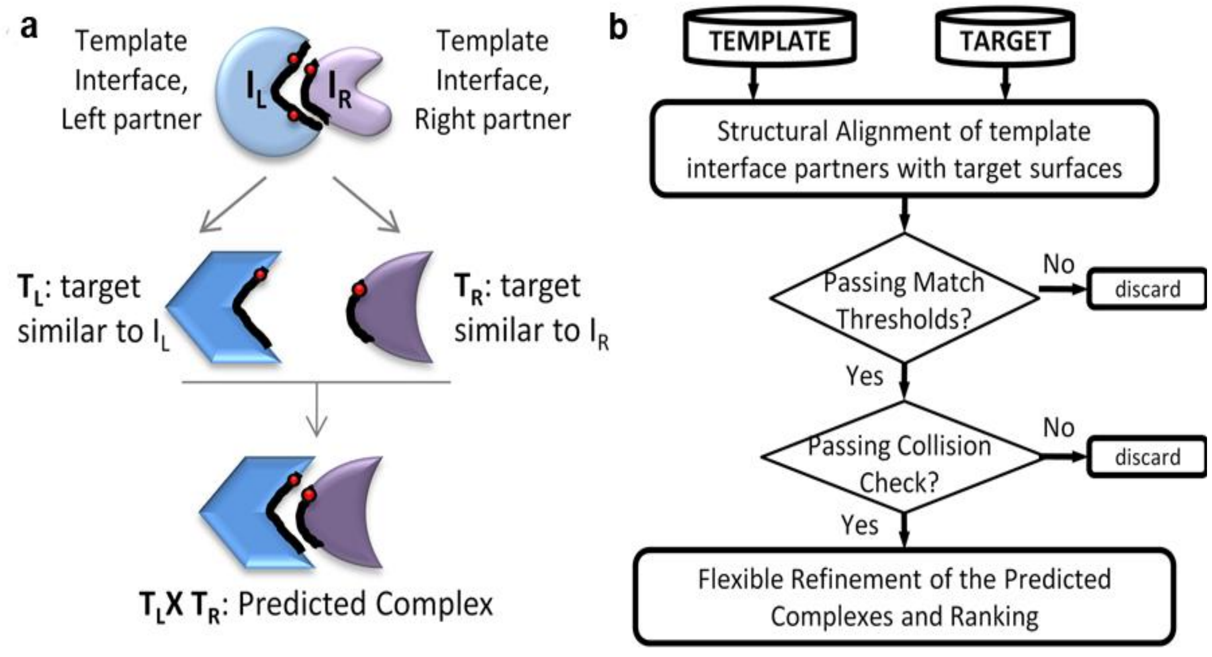

a
Template Interface, Left partner
I_L
I_R
Template Interface, Right partner
T_L: target similar to I_L
T_R: target similar to I_R
T_LX T_R: Predicted Complex
b
TEMPLATE
TARGET
Structural Alignment of template interface partners with target surfaces
Passing Match Thresholds?
No
discard
Yes
Passing Collision Check?
No
discard
Yes
Flexible Refinement of the Predicted Complexes and Ranking

Figure 3.3 The algorithm of prism [38]

In the surface extraction step, the Naccess program [39] is used. It is a method based on the idea of rolling a probe (solvent molecule) of given size on a van der Waals surface. The accessible surface area (ASA) of an atom is defined as the area on the surface of a sphere of radius *R*, which is given by the sum of the van der Waals radius of the atom and the chosen radius of the solvent molecule.

MultiProt [36, 37] is a program that evaluates the multiple structural alignments of proteins. It finds the common geometrical cores between the input molecules. Unlike most of the existing methods, MultiProt does not require that all the input molecules participate in the alignment. Actually, it efficiently detects high scoring partial multiple alignments for all possible number of molecules from the input. The final structural alignment can either preserve the sequence order (like sequence alignment), or be sequence order independent.

To rank the predicted complexes, FiberDock (an efficient method for flexible refinement and rescoring of rigid-body protein-protein docking solutions) [34, 40] is used. The method models both side-chain and backbone flexibility and performs rigid body optimization on the ligand orientation. The backbone and side-chain movements are modeled according to the binding van der Waals forces between the receptor and ligand. The method is able to model both global and local conformational changes, such as opening of binding sites and loop movement. After refining all the docking solution candidates, the refined models are re-scored according to an energy function.

3.1.4 PRISM Protocol

PRISM Protocol is a downloadable protocol which uses PRISM algorithm to find interactions between proteins. It is available at http://prism.ccbb.ku.edu.tr/prism_protocol/ . It is a zip package consisting collections of programs which include Python scripts, some external tools like MultiProt, FiberDock. It predicts interactions between a set of proteins in four steps as in the PRISM algorithm. For these four steps, there are four different folders and four different python scripts.

Before beginning to run PRISM protocol, users should install some external programs in the respective folder by following their installation guides and obtain the executables of these programs. After that, users decide whether they want to use the default template set or a customized template set to predict interactions between their target proteins. The default template set is directly available in the PRISM protocol package. If users want to use their own template set, they should construct it according to the specifications. There are python scripts available also for this generation process. After the template set is ready, users prepare their target protein structures. Target protein structure files in PDB format can be downloaded manually or a python script can be run to download them automatically by listing them in a file.

After the preparation of template and target structures, the protocol can be run by executing four python scripts for the four steps in the algorithm. In the first phase, users go to the respective directory for the extraction of surfaces of target proteins. After putting the target protein list file into the directory, related python script is run with different parameters (working folders, number of target proteins etc.). When the surface extraction process is finished, a file which contains the surface files' paths should be copied to another folder for the next step. In the second phase, a phyton script for structural alignments between template interfaces and target structures is run. This script also needs some parameters. In the third phase, another script is run to transform target proteins into a complex. In this step, if the template set in use is not the default set, some replacements should be done with the files created in the template preparation step before running the script. In the last step, flexible refinement of docking solutions is done using a phyton script. When this step is finished running, results will be available as text files in which every column indicate a value. Users should be familiar with the meaning of each column to interpret the results.

3.2 PRISM's Web-Database Application Architecture (4-tier)

Many web-database applications use 3-tier application architecture as discussed in Section 2.4. This architecture is also used in the implementation of PRISM web server with a slight difference. In PRISM architecture, the middle layer (logic tier) is divided into two layers as application logic and compute server components. Therefore, it can be considered as a 4-tier architecture. Visual diagram of the architecture can be seen in Figure 3.4.

As discussed in the previous section, it takes a lot of effort to run the PRISM algorithm from the beginning to the end. In addition, computation of the algorithm can take a long time according to the number of target proteins and the template set used. Since calculation takes a lot of time, there is a bottleneck in the middle layer of the architecture. To eliminate this bottleneck, computation part is separated from the application logic. The communication between these two components is provided by an interprocess message queue system explained in Section 2.5.

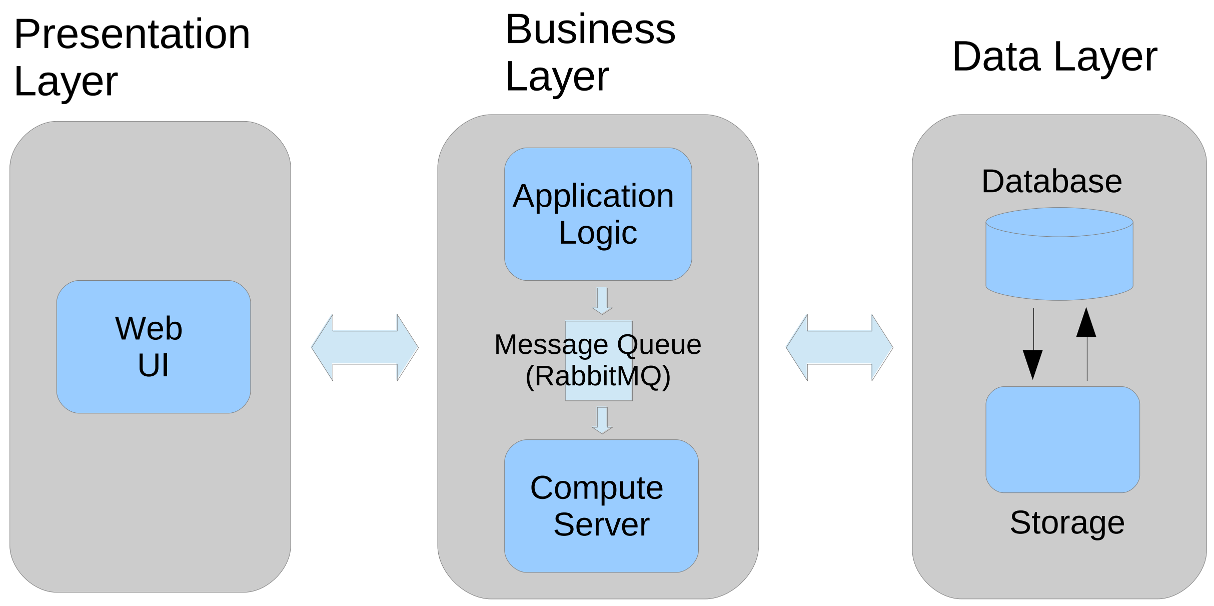


Figure 3.4 Web-Database Application Architecture of PRISM web server. Note that Business Layer is separated into two parts as Application Logic and Compute Server.

In the implementation of PRISM web server, RabbitMQ (See Section 2.5) is used. Application logic speaks with the compute server through RabbitMQ. This is done as follows: A phyton program sends messages to another python program through a queue named “prism”. The other program receives messages from the queue, interpret these messages and create jobs to send to the compute server. Jobs on the compute server are handled by Sun Grid Engine which is an open source batch-queuing system.

This process takes place when the entered target proteins is not in our target dataset and prism needs to be run from the beginning. In this case, computation takes a lot of time. There are four steps and four different scripts to be run to predict the interactions between input proteins. Data handling at these steps of PRISM is also hard. Manual operations between these steps (see Section 3.1.4) are automated using the web server. The folders needed are created using unique job ids to run more than one job concurrently.

The presentation layer and the data layer of the PRISM web server is similar to the typical 3-tier web-database application architectures. Presentation layer includes the web user interface. Front end users interact with the web server using this layer. After communicating the middle layer, presentation layer presents results to the users. Data layer includes the database and the storage (in PRISM's case file system). This layer transmits information from the database to the middle layer.

Chapter 4

RESULTS

4.1 Design of PRISM

PRISM's architecture is a typical multitier (n-tier) web application architecture, which is a 4-tier web application in our case as discussed in Section 3.2. The main functionality of the PRISM web server is to provide to users to run the PRISM protocol automatically without doing any manual operations. Other than that, contents of a relational database (PRISM database) can be queired via WWW(World Wide Web) [41].

In the implementation, HTML, PHP, MySQL and Phyton technologies are used. To create the web pages in PRISM web server, HTML(HyperText Markup Language) is used. HTML is the main markup language for creating web pages and other information that can be displayed in a web browser [42]. The World Wide Web is composed primarily of HTML documents transmitted from web servers to web browsers. PHP is also used as embedded into HTML in PRISM web server. PHP(HyperText Preprocessor) is an open-source scripting language commonly used in web development [43]. It can be used on server-side scripting, command line scripting and creating desktop applications but its main area of use is for developing server-side applications. PHP supports all major operating systems and wide range of databases including MySQL and this has an influence of choosing the PHP as a programming language in the implementation of PRISM.

MySQL is used in the database component of PRISM web server. MySQL is the most popular relational database management system [44]. The SQL part of 'MySQL' stands for 'Structured Query Language'. SQL is the most commonly used language to access databases. MySQL provides multi-user access to more than one databases. In PRISM web application, SQL statements are embedded into PHP codes which means that users do not directly access to the database. They query the contents of the database via the web server which is explained in Section 4.1.1. However, administrators can access and update the database locally. For these administrative purposes, Phyton scripts are used. Phyton is a object-oriented, high-level programing language with dynamic semantics (dynamic typing and binding) [45].

4.1.1 Database Design

All information about template structures, target structures and predictions are kept in a relational database. Using relational database to store information has some advantages. The most crucial one is that it is easy to interpret information when it is stored in tables consisting of rows and columns. Also, it is easy to manipulate data to give information in the form which it is desired. The database of PRISM is implemented using the MySQL system as mentioned in the previous section and its current version is MySQL 5.5.31. It consists of 6 tables and the ER diagram of these tables can be seen in Figure 4.1.

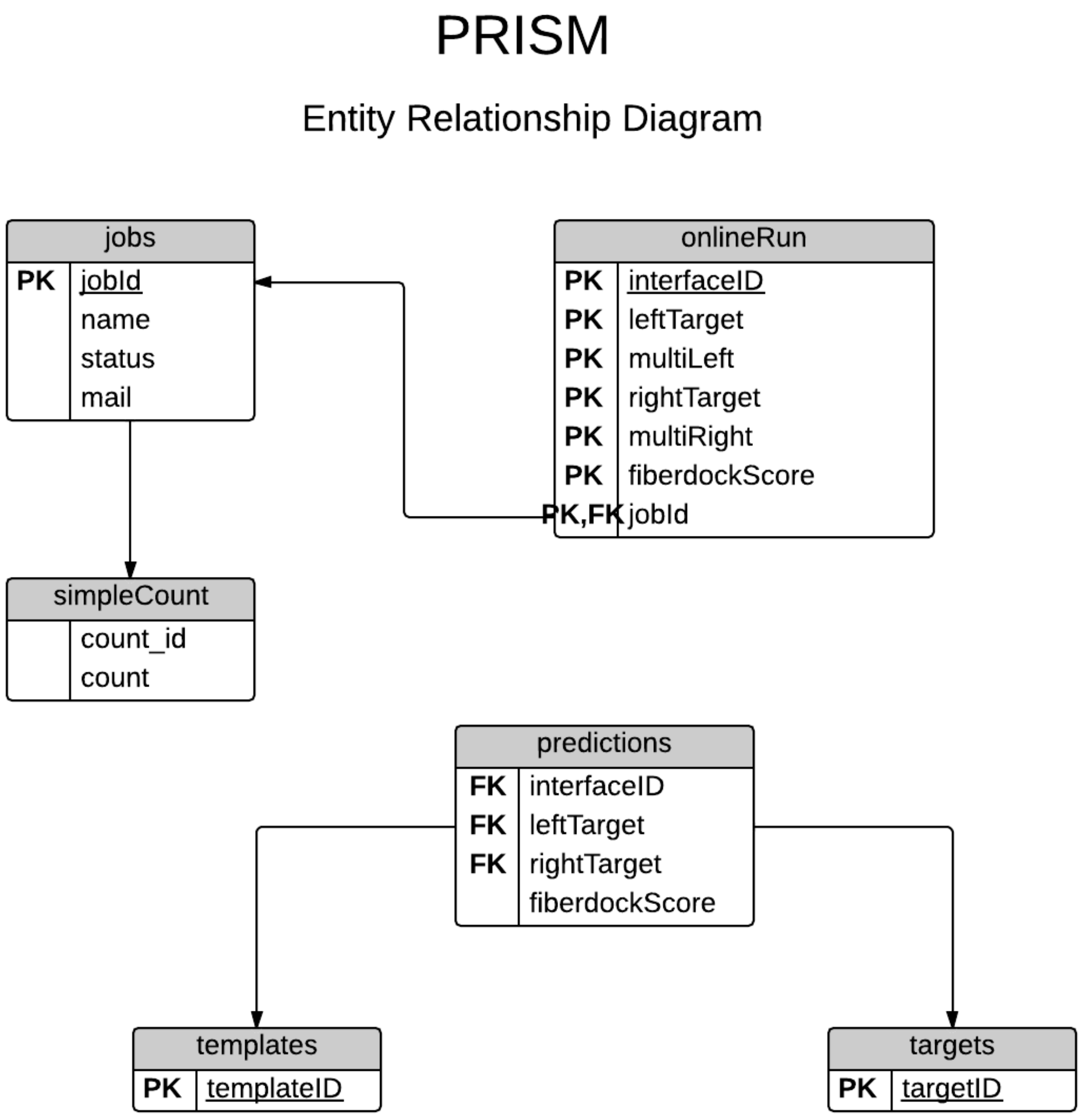


Figure 4.1 ER Diagram of PRISM Database

The properties of each column of the tables is explained below.

Field: The name of the corresponding column in the database table.

Type: Describes the type of individual values the column can have. The numbers in parantheses indicate the size allocated in bytes for each value.

Key: Shows whether or not the column acts as part of a primary key (PK) or foreign key (FK) for that table.

Default: The default value assigned to that column if one is not set explicitly.

Extra: Miscellaneous information

The details of tables present in the PRISM database will be presented next.

4.1.1.1 Templates Table

Template dataset of PRISM (see Section 3.1.1) is stored in 'templates' table whose details can be seen in Table 4.1. This includes only one column which keeps the 'templateID' information of these template proteins. 'templateID' is a field which is six characters long and it consists of four-character PDB code followed by two-character chain identifiers of the interface on that template structure (e.g. 1c4zAD).

Table 4.1 Description of 'templates' table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
templateID	varchar(6)	NO	PK		

4.1.1.2 Targets Table

Target datasef of PRISM (see Section 3.1.2) is stored in 'targets' table whose details can be seen in Table 4.2. This table also includes just one column like templates table. This column is 'targetID' column and it keeps a unique id for the target protein structures. 'targetID' can take two forms: First one is a four-character PDB code in case of monomers with no chain identifiers (e.g. 1bor). Second one is a five-character code which includes a chain identifier as a fifth character appended to the four-character PDB code (e.g. 1d5fA).

Table 4.2 Description of 'targets' table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
targetID	varchar(6)	NO	PK		

4.1.1.3 Predictions Table

This table keeps the current prediction results which are calculated among the target protein structures in the targets table and using the template protein structures in the templates table. ‘interfaceID’ column represent the interface on the template protein via which the target proteins interact. ‘leftTarget’ is the target protein structure matching to the the left partner of the template and ‘rightTarget’ is the target protein structure matching to the right partner of the template. ‘fiberdockScore’ column is the calculated score from the flexible refinement of the predicted complexes with the template and target proteins. Description of Predictions table can be seen in Table 4.3.

Table 4.3 Description of ‘predictions’ table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
interfaceID	varchar(6)	YES		NULL	
leftTarget	varchar(10)	YES		NULL	
rightTarget	varchar(10)	YES		NULL	
fiberdockScore	float	YES		NULL	

4.1.1.4 Jobs Table

When the user wants to see possible interactions between target proteins and these proteins are not in our target dataset, PRISM algorithm runs from the beginning for these protein structures as explained in Section 4.1.3.3 Predictions Section These runs are added to a queue using RabbitMQ (see Section 2.5).

The contents of ‘jobs’ table consists the jobs which are added to the queue. These jobs can be running at that moment or finished running. ‘status’ column stores the current status of the jobs. It is ‘0’ if the job is running at that moment, and ‘1’ if the job is finished already. ‘jobId’ column stores the id of the jobs which are given to the users when they submit the job. It is given incrementally by looking at the ‘SimpleCount’ table. ‘name’ column includes the name of the job that the user chooses. It can be null since the users give a name to their submitted job if they want to (it is optional). ‘mail’ column holds the email adresses that users give when they are submitting a job. To retrieve a job, this email adress and the job id are checked to see whether they match or not from this table. Description of 'jobs' table and the properties of its columns can be seen in Table 4.4.

Table 4.4 Description of 'jobs' table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
jobId	varchar(20)	NO	PRI		
name	varchar(20)	YES		NULL	
status	int(11)	YES		NULL	
mail	varchar(50)	YES		NULL	

4.1.1.5 OnlineRun Table

This table contains the prediction results which are calculated when the target structures given by the user are not in our dataset. 'leftTarget' column represents the first protein and 'rightTarget' column represents the second protein given by the user. Actually, 'leftTarget' column is for the target protein structure which matches with the left side of the interface and 'rightTarget' column is for the target protein structure which matches with the right side of the interface via which they interact. 'multiLeft' is the MultiProt solution number for the alignment between the template protein and the target protein matching to the left partner of this template. Likewise, 'multiRight' is the MultiProt solution number for the alignment between the template protein and the target protein matching to the right partner of this template. 'fiberdockScore' is the calculated global energy for this predicted complex. 'jobId' column represents the id for this particular job which is given to the user when the job is first submitted. It is a foreign key column and it refers to the 'jobId' column of 'jobs' table.

Table 4.5 Description of ‘onlineRun’ table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
interfaceID	varchar(6)	NO	PRI		
leftTarget	varchar(10)	NO	PRI		
multiLeft	int(11)	NO	PRI	0	
rightTarget	varchar(10)	NO	PRI		
multiRight	int(11)	NO	PRI	0	
fiberdockScore	float	NO	PRI	0	
jobID	varchar(20)	NO	PRI		

4.1.1.6 SimpleCount Table

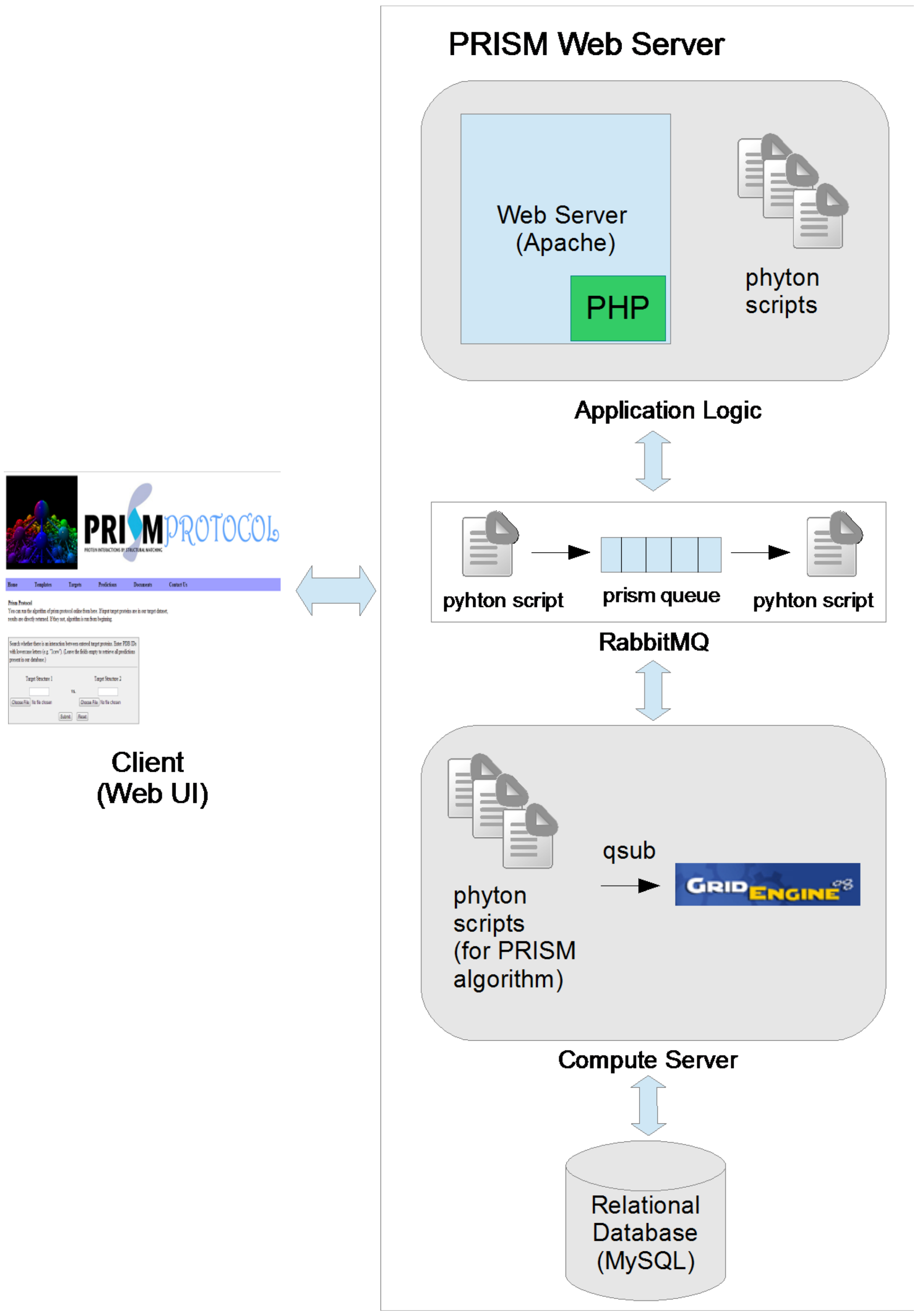
It is a simple table which keeps the last job id that is given to the users to be able to determine the new job’s id. If a new job is submitted by the users, it is given an id which is one more than this last id.

Table 4.6 Description of ‘simpleCount’ table

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
count_id	int(11)	YES		NULL	
count	int(11)	YES		NULL	

4.1.2 Application Architecture Design

PRISM’s application architecture is a 4-tier architecture as discussed in Section 3.2. In a 4-tier architecture, all of the data storage and retrieval processes are logically and usually physically located on a single tier. A 4-tier architecture allows an unlimited number of programs to run simultaneously, send information to one another, use different protocols to communicate, and interact concurrently. This allows for a much more powerful application, providing many different services to many different clients. The details of the architecture can be examined from Figure 4.2.



Prism Protocol
You can run the algorithm of prism protocol online from here. If input target proteins are in our target dataset, results are directly returned. If they not, algorithm is run from begining.

Search whether there is an interaction between entered target proteins. Enter PDB IDs with lowercase letters (e.g. "1cow"). (Leave the fields empty to retrieve all predictions present in our database.)

Figure 4.2 Application Architecture Design of PRISM Web Server

In PRISM's application architecture, the middle layer is divided into two parts as application logic and compute server as seen in the figure above. It is done to eliminate the bottleneck in this layer due to the difficulty of computing the PRISM algorithm. Application logic determines the reasoning to make the data meaningfully available for the presentation layer. It also processes the dynamic content (e.g. showing existing predictions) using PHP scripts. On the other hand, compute server takes place between application logic and data layer. It handles the database operations which load too much burden to the application logic. By this, database transactions would have been separated from the application logic and encapsulated in the compute server.

Communication between application logic and compute server is handled by an interprocess message queue system namely RabbitMQ (see Section 2.5). This is done as explained in Section 3.2: A phyton program sends messages to another python program through a queue named "prism". The other program receives messages from the queue and interpret these messages. The interpreration occurs like this: This program checks the jobs in the 'jobs' table (see Section 4.1.1.4) on the database. It creates shell scripts ('run.sh' files) for each ongoing job on this table. With these scripts, the jobs are created to send to the compute server. Jobs on the compute server are submitted to Sun Grid Engine, which is an open source batch-queuing system, using qsub command.

Dividing the middle layer as application logic and compute server has some advantages. Even if it is not implemented in current version of PRISM web server, recovering erroneous jobs running with qsub command is more applicable. Using qsub command, keeping track of possible errors and monitor them is easy to implement.

After the steps of PRISM algorithm finish, a phyton program is called to add the results to the database. To be able to understand whether a job is finished running or not, 'status' column is checked from the 'jobs' table with the job id as discussed in Section 4.1.1.4. 'onlineRun' table (see Section 4.1.1.5) can also be checked with the target proteins in hand. If the results are available, these results are directly returned to the users. If there are no rows in the 'onlineRun' table for these target proteins, we understand that the job is stil running and users are also told so.

4.1.3 Web Interface Design

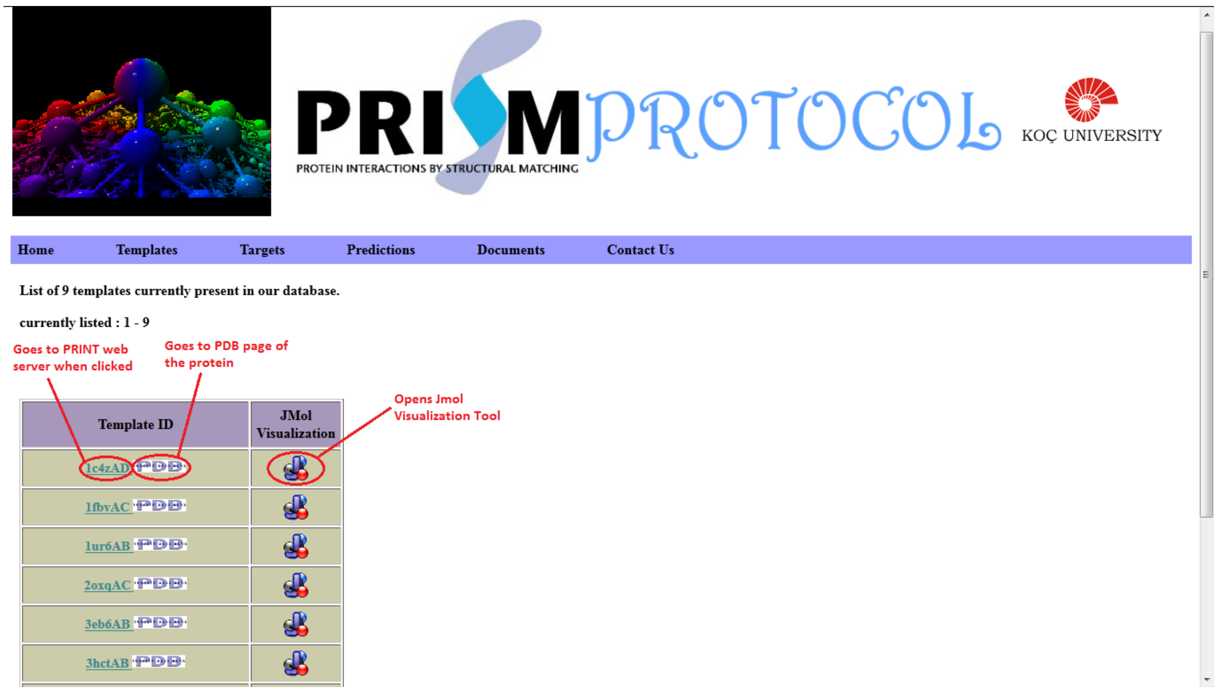
PRISM can be accessed through a web site http://gene.ccbb.ku.edu.tr/protocol. All the pages of the PRISM web site are served via the PHP module of Apache server version 2.2.22.

PRISM protocol, a downloadable protocol which is used to run PRISM algorithm, requires users to setup necessary environments, install some external tools, prepare the datasets and run the algorithm step by step (as discussed in Section 3.1.4). Between these steps, users also have to set up the working folders, copy some files to different folders etc.

The main goal of the PRISM web server is to eliminate these steps in the protocol and provide users to run the algoritm just by entering PDB ids of target protein structures. Other than that, users also can access the contents of PRISM database by using different sections of PRISM web server. The main sections of the web site are templates, targets and predictions sections. These sections will be presented next.

4.1.3.1 Templates Section

Contents of the template dataset is available in the Templates section as seen in Figure 4.3. Users can browse through the list of templates available in the PRISM database through this section. Also, they can go to the protein data bank (PDB) web site to see the details of a template when PDB id of one of the templates under the Template ID section is clicked. Likewise, users can go to the PRINT web server to see the details of the interface (see Section 2.1) of a protein when they click to the PDB image next to the PDB ID of the protein if an interface exists in the protein structure.


PRISM PROTOCOL
PROTEIN INTERACTIONS BY STRUCTURAL MATCHING
KOÇ UNIVERSITY
Home
Templates
Targets
Predictions
Documents
Contact Us
Goes to PRINT web server when clicked
Goes to PDB page of the protein
Opens Jmol Visualization Tool

List of 9 templates currently present in our database.

currently listed : 1 - 9

Template ID	JMol Visualization
1c4zAD PDB	
1fbvAC PDB	
1ur6AB PDB	
2oxqAC PDB	
3eb6AB PDB	
3hctAB PDB	

Figure 4.3 Templates Page

In this section, users also can visualize the template structures using Jmol plug-in as seen in Figure 4.4. Jmol is a open-source Java viewer for chemical structures in 3D [25, 46]. It has three versions for usage: 1) Jmol Applet, which is integrated into web pages as a web browser applet, 2) Jmol application, downloadable as a desktop application, 3) Jmol Viewer, a development tool kit that can be integrated to other Java applications. In PRISM web server, the Jmol Applet is used. If the user clicks on the Jmol logo under the Jmol visualization section, a java applet in a seperate window opens and a visualization of the chosen template structure is presented to the user. Since Jmol Applet works as a java applet, Java software must be installed to the computer and Java applets must be given permission to run. Jmol can work on all major web browsers like Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera etc.

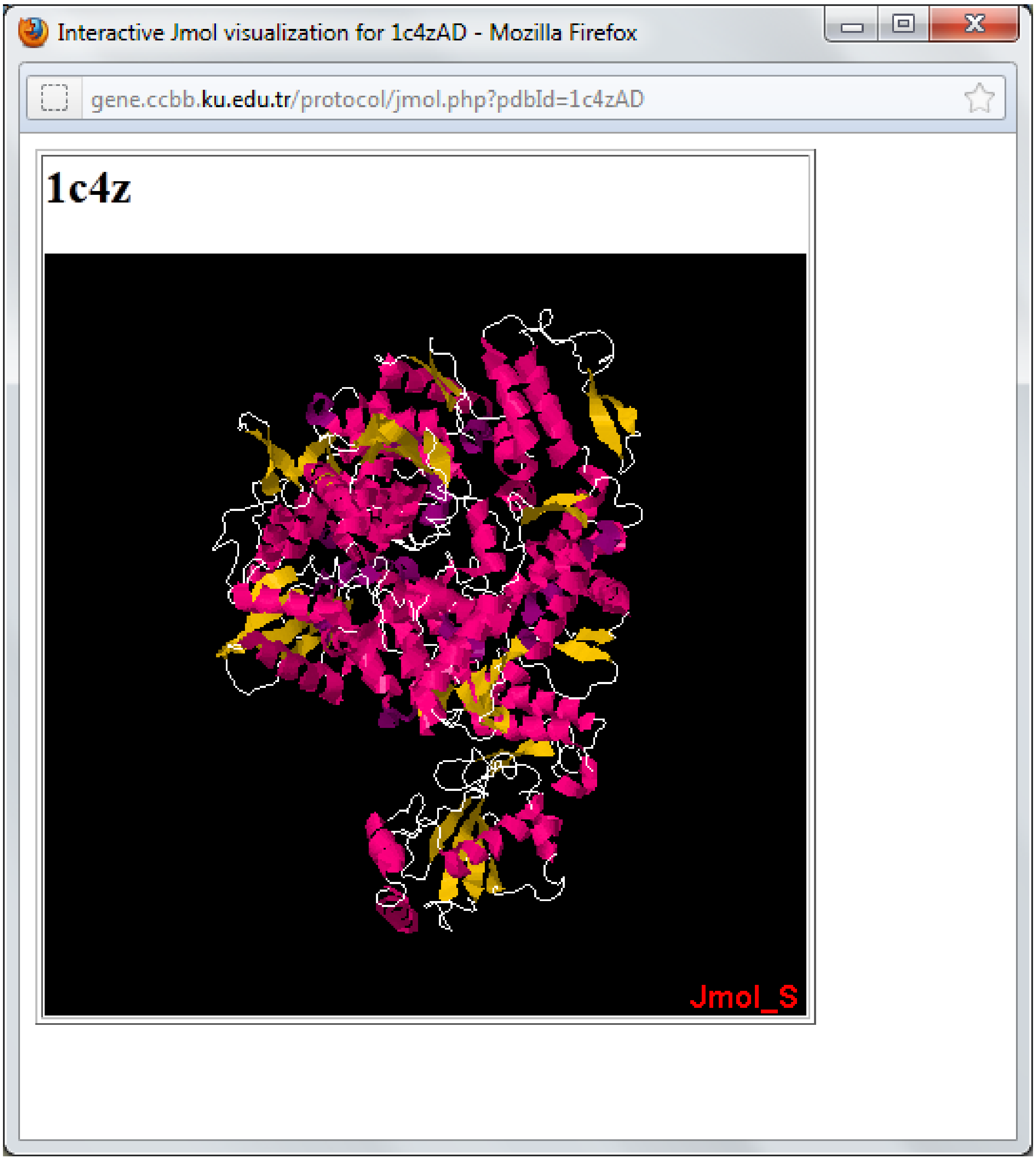
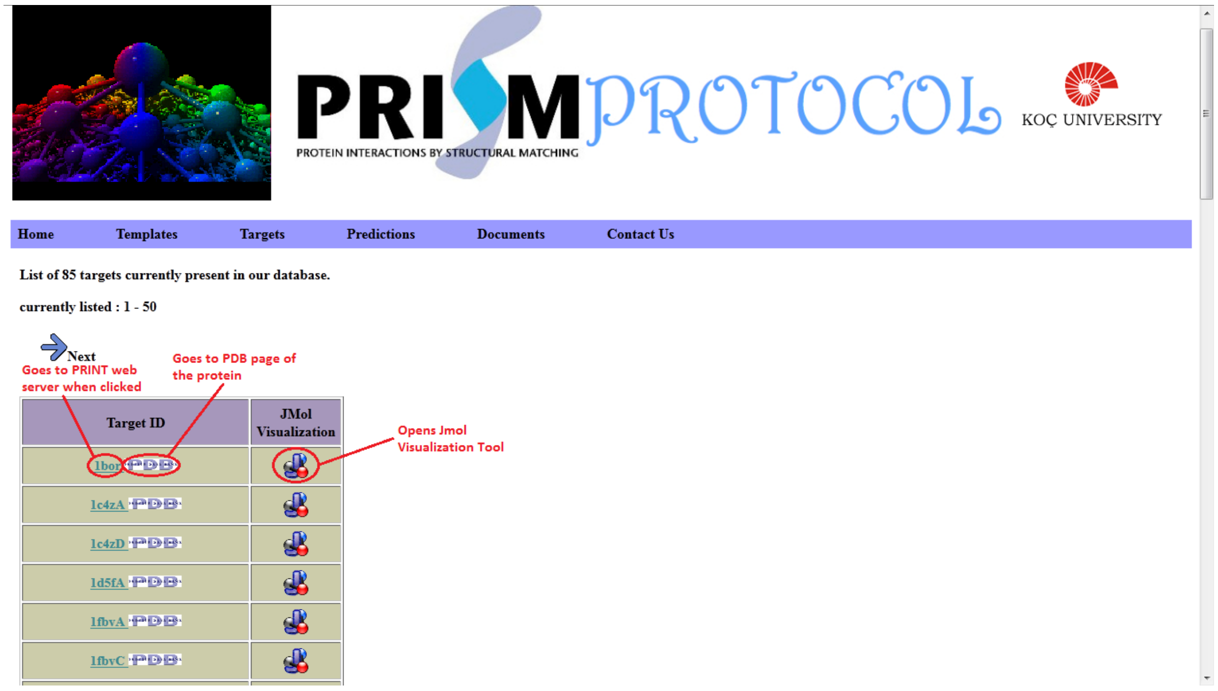

Interactive Jmol visualization for 1c4zAD - Mozilla Firefox
gene.ccbb.ku.edu.tr/protocol/jmol.php?pdbId=1c4zAD
1c4z
Jmol_S

Figure 4.4 Jmol Visualization of a Template (1c4z)

4.1.3.2 Targets Section

Figure 4.5 shows the Targets section which works similar to the Templates section, but this time users access the target structures stored in the PRISM database. Once a target protein is selected among the results, the details of the target structure in the protein data bank (PDB) and PRINT web server can be viewed in a similar way to the Templates section.


PRISM PROTOCOL
PROTEIN INTERACTIONS BY STRUCTURAL MATCHING
KOÇ UNIVERSITY
Home
Templates
Targets
Predictions
Documents
Contact Us
Next
Goes to PRINT web server when clicked
Goes to PDB page of the protein
Opens Jmol Visualization Tool

List of 85 targets currently present in our database.

currently listed : 1 - 50

Target ID	JMol Visualization
1bor PDB	
1c4zA PDB	
1c4zD PDB	
1d5fA PDB	
1fbvA PDB	
1fbvC PDB	

Figure 4.5 Targets Page

Jmol visualization tool also works as in the Templates section. If the user clicks on the Jmol image under the Jmol visualization section, an applet opens and shows the 3D representation of the target protein. Figure 4.6 shows such an example.

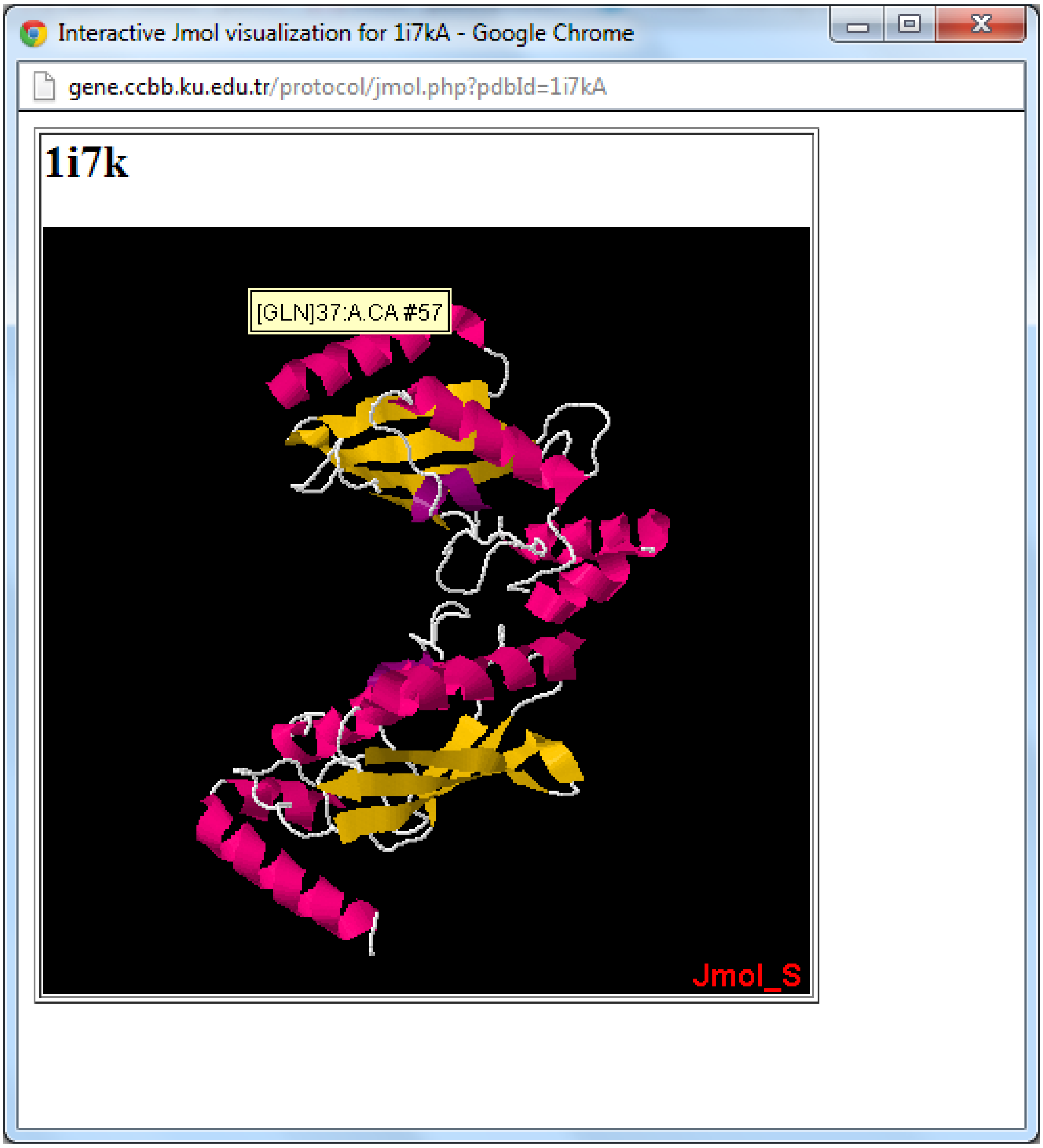

Interactive Jmol visualization for 1i7kA - Google Chrome
gene.ccbb.ku.edu.tr/protocol/jmol.php?pdbId=1i7kA
1i7k
[GLN]37:A.CA #57
Jmol_S

Figure 4.6 Visualization of a Target (1i7k)

4.1.3.3 Predictions Section

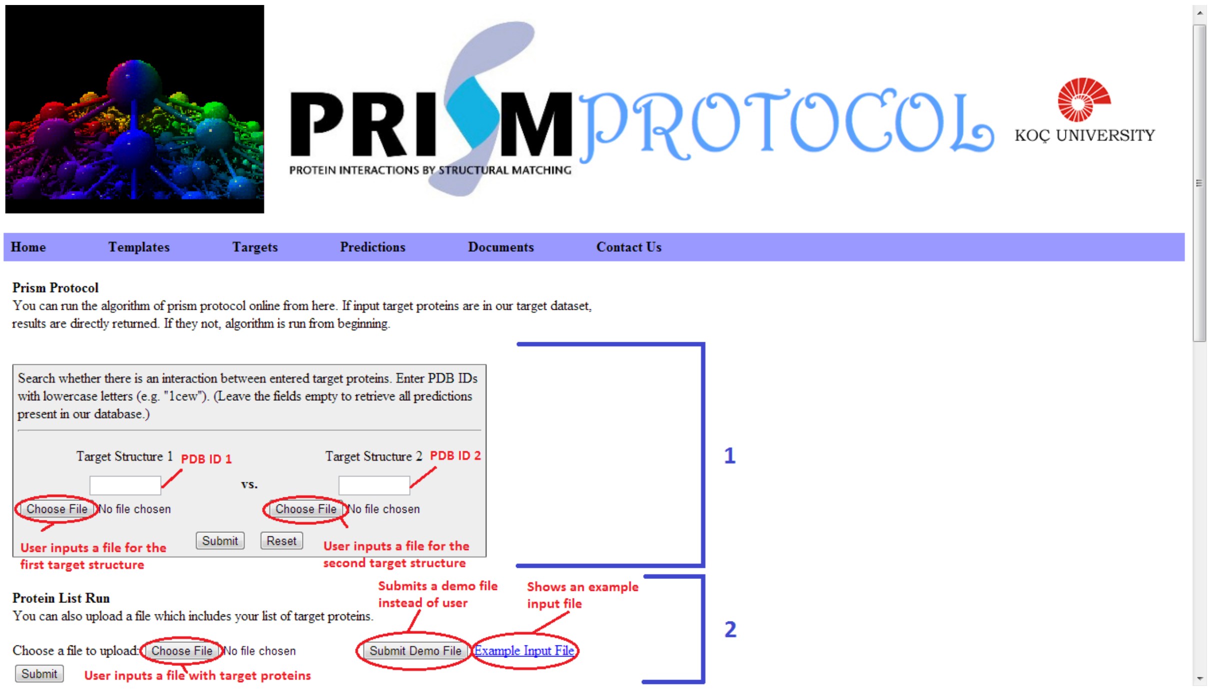
In the Predictions section, users are basically provided the prediction results stored in the database and can search for possible interactions between target structures. However, there are cases in which the precalculated interaction predictions on the target dataset do not cover the protein in hand. In these situations, the interactions can be computed online by running the prediction algorithm. In this process, the query structure splits into its constituent chains, the homologies among them are eliminated, and the algorithm is run for each distinct chain.

The basic operations in the Predictions part of the web server is to find interactions between target proteins (two or more), if these interactions are not present in the database, then to run PRISM algorithm from the beginning for the target proteins and to retrieve results which are calculated earlier on the web server. There are five main parts in the Predictions section:

Interaction Between Two Target Proteins

The first part of the Predictions section is to view the interactions between two target proteins (see Figure 4.7). Users can enter two PDB IDs for the target structures that are wanted to examine in the respective fields. Instead of entering PDB IDs, users also can upload two files which include protein structures that are wanted to examine when the structures are not present in PDB or users do not have the PDB IDs. After entering the two target structures (in either way), user clicks the submit button. If entered target proteins are in our target dataset, interactions between them which are present in our database directly returned to the users. If they are not in our target dataset, PRISM algorithm is run from the beginning and results are returned to the users whenever they are finished as seen in the Results page (see Figure 4.10). If users do not enter any PDB IDs and click to the submit button, all prediction results in the PRISM database are returned to the users as seen in the Figure 4.8.

In the result page, the two target structures which interacts and the template structure they are interacting via is viewed. Also, the FiberDock score of the predicted complexes can be seen in this page. Users can also go to PDB or PRINT web sites of the templates and targets as in the Templates and Targets sections of PRISM web server. Jmol visualization tool also works in the same way as in the Templates and Targets sections.



Prism Protocol
You can run the algorithm of prism protocol online from here. If input target proteins are in our target dataset, results are directly returned. If they not, algorithm is run from beginning.

Search whether there is an interaction between entered target proteins. Enter PDB IDs with lowercase letters (e.g. "1cew"). (Leave the fields empty to retrieve all predictions present in our database.)

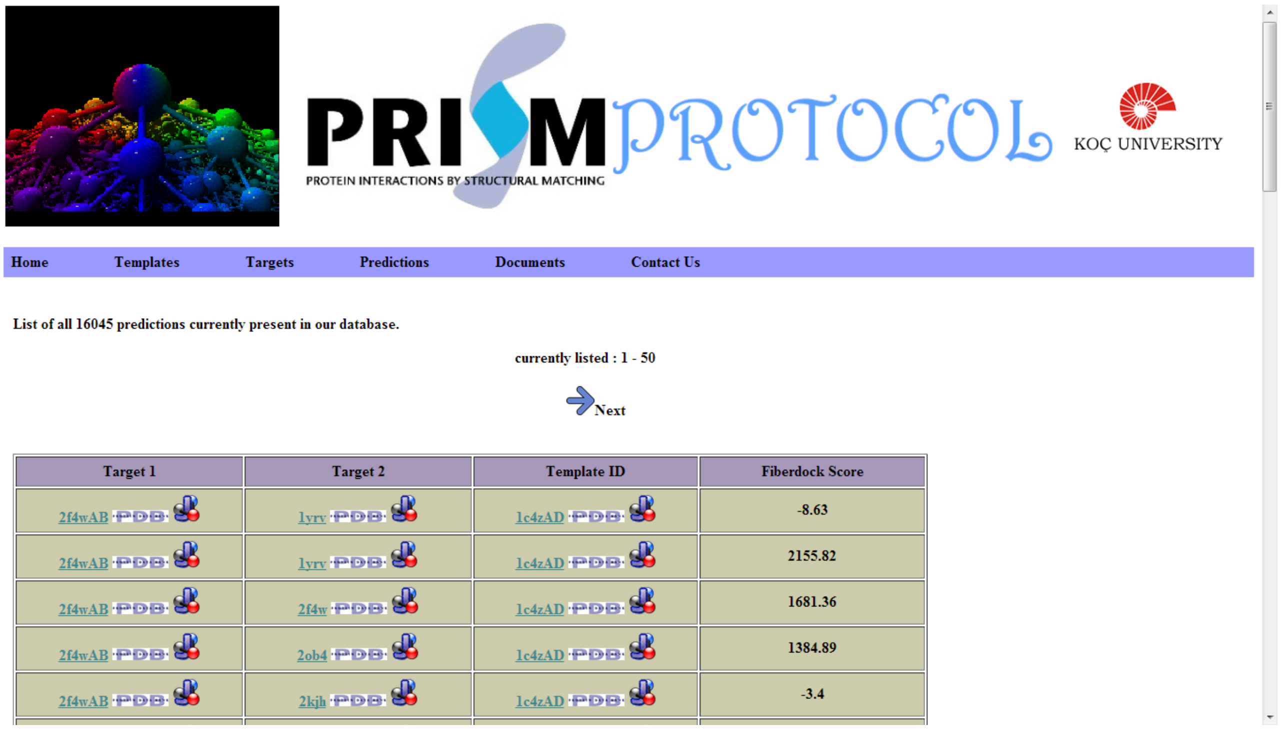
Protein List Run
You can also upload a file which includes your list of target proteins.

Figure 4.7 First two sections of the Predictions page

Protein List Run

Second part of the Predictions section is the Protein List Run part (see Figure 4.7). In this part, users can enter more than two target proteins in a file which includes PDB IDs of the target structures that are wanted to examine. An example of such a file can be viewed if the users click on the Example Input File link on the rightmost part. It is a file which includes PDB ID of a protein in each line. Also, this example file can be submitted as a demo file if Submit Demo File button is clicked next to the example input file link.

After the file is submitted (whether the demo file or user's own file), PRISM protocol is run from the beginning and try to find interactions between the target proteins in every combination of two. When the interactions are found, they are returned to the users as seen in the Results page (see Figure 4.10).



List of all 16045 predictions currently present in our database.

currently listed : 1 - 50

Target 1	Target 2	Template ID	Fiberdock Score
2f4wAB PDB	1yrv PDB	1c4zAD PDB	-8.63
2f4wAB PDB	1yrv PDB	1c4zAD PDB	2155.82
2f4wAB PDB	2f4w PDB	1c4zAD PDB	1681.36
2f4wAB PDB	2ob4 PDB	1c4zAD PDB	1384.89
2f4wAB PDB	2kjh PDB	1c4zAD PDB	-3.4

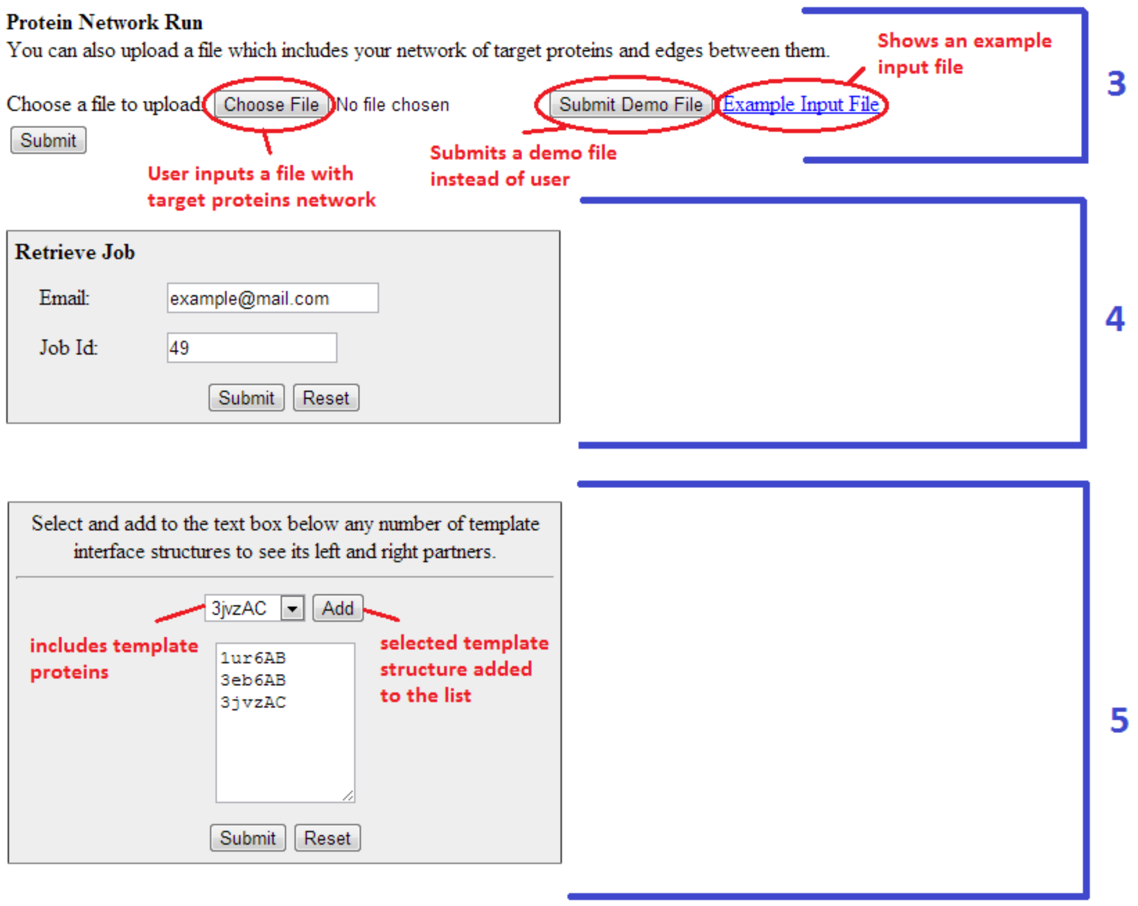
Figure 4.8 Page of all Predictions present in PRISM

Protein Network Run

Third part of the Predictions section is the Protein Network Run part (see Figure 4.9). This part is similar to the Protein List Run part which is explained in the previous section. The difference between them is that users enter network of proteins instead of list of proteins this time. Users prepare the input files to include in groups of two target protein structures. In the input file, every line includes two proteins and PRISM algorithm examines the interactions between these two proteins line by line.

As in the Protein List Run part, users can also view an example input file by clicking the Example Input File link and submit this example file as a demo by clicking the Submit Demo File button.

After the file is submitted (whether the demo file or user's own file), PRISM protocol is run from the beginning and try to find interactions between the target proteins as in the protein list run discussed in the previous section. But this time only the interactions between the proteins given as groups of two are searched instead of all binary combinations examined. When the interactions are found, they are returned to the users as seen in the Results page (see Figure 4.10).


Protein Network Run

You can also upload a file which includes your network of target proteins and edges between them.

Choose a file to upload: Choose File No file chosen

Submit Demo File Example Input File

Submit

Retrieve Job

Email: example@mail.com

Job Id: 49

Submit Reset

Select and add to the text box below any number of template interface structures to see its left and right partners.

3jvzAC Add

```
1ur6AB
3eb6AB
3jvzAC
```

Submit Reset

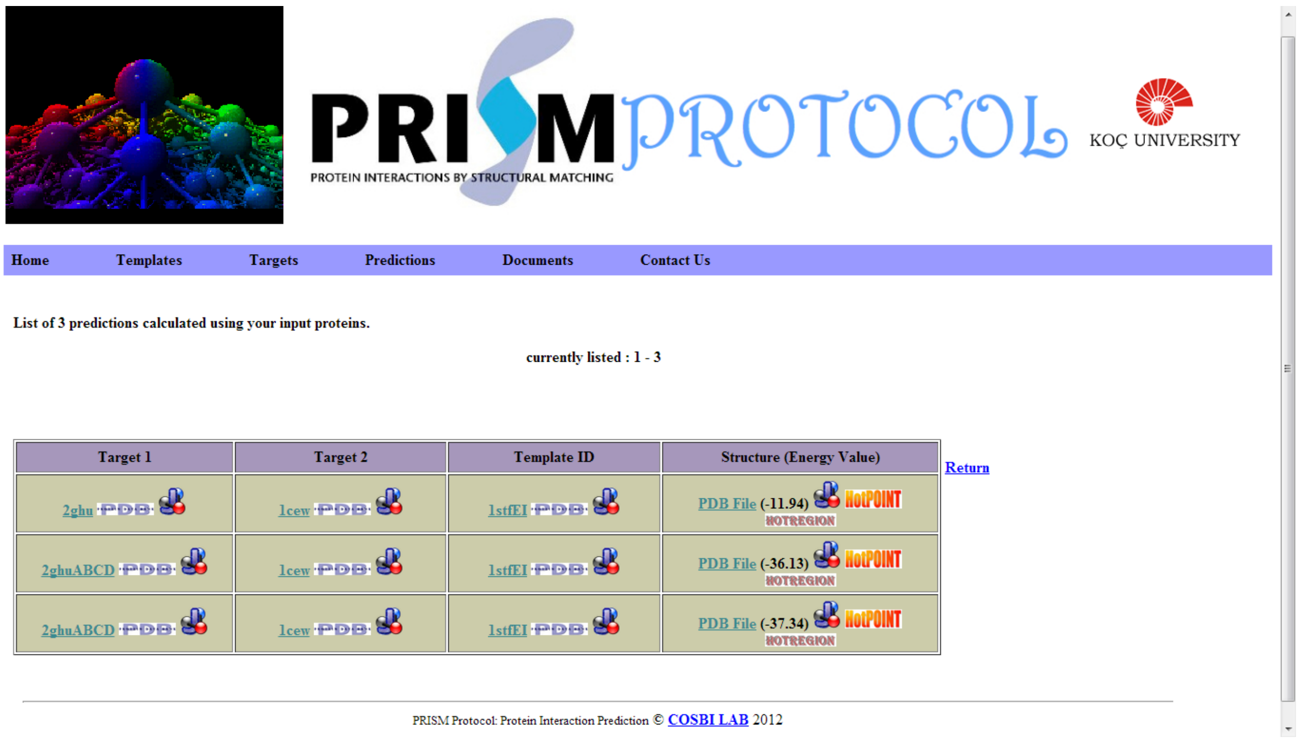
Shows an example input file
User inputs a file with target proteins network
Submits a demo file instead of user
includes template proteins
selected template structure added to the list
3
4
5

Figure 4.9 Last three sections of the Predictions page

Retrieve Job

Fourth part of the Predictions section is the Retrieve Job part (see Figure 4.9). In this part, users can query the previous jobs that they submitted to the server. To retrieve the old jobs, users enter their email adresses that they entered when they were submitting the job they want to retrieve now and the id of the jab that they submitted which is given to them by the server.

If email or job id is not entered, an alert window opens which says to users that they should enter them. If the entered job id and email do not match, users are confronted with a page which states this mismatch and directs the users to return the previous page. If the job that is wanted to retrieve is stil running and the results are not ready, users are confronted with a page that states that protocol is running for the job with the job id entered and directs the users to return the previous page similarly. If the job is finished and results are ready, they are returned to the users as seen in the Results page (see Figure 4.10).



List of 3 predictions calculated using your input proteins.

currently listed : 1 - 3

Target 1	Target 2	Template ID	Structure (Energy Value)
2ghu PDB	1cew PDB	1stfEI PDB	PDB File (-11.94) HotPOINT HOTREGION
2ghuABCD PDB	1cew PDB	1stfEI PDB	PDB File (-36.13) HotPOINT HOTREGION
2ghuABCD PDB	1cew PDB	1stfEI PDB	PDB File (-37.34) HotPOINT HOTREGION

Return

PRISM Protocol: Protein Interaction Prediction © COSBI LAB 2012

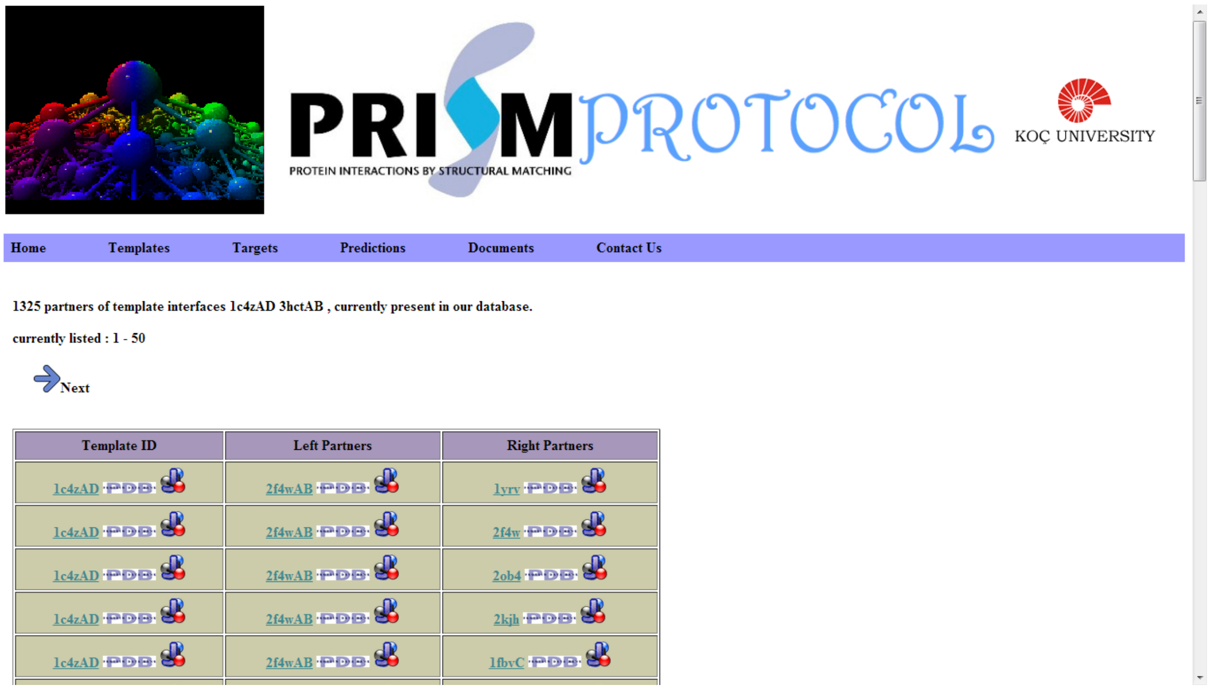
Figure 4.10 Results Page

Left and Right Partners of Template Structures

Last part of the Predictions sections is the part where the users can search one or more template protein structures' left and right partners (see Figure 4.9). It means that users can view the target protein structures which interact via the chosen template structures.

In this part, all templates present in the PRISM database is listed in a combobox. Users can choose the template structures whose partners they want to view and add to the list below by clicking the Add button. After finishing to add the template structures, users click the submit button to see the results.

After all chosen template structures are submitted to the system, users are confronted with a page (see Figure 4.11) where all left and right partners are listed for the chosen templates. These structures can be viewed in PDB or PRINT web sites as in the Templates or Targets sections. Also, they can be visualized using Jmol similarly.



1325 partners of template interfaces 1c4zAD 3hctAB , currently present in our database.

currently listed : 1 - 50

Template ID	Left Partners	Right Partners
1c4zAD PDB	2f4wAB PDB	1yrv PDB
1c4zAD PDB	2f4wAB PDB	2f4w PDB
1c4zAD PDB	2f4wAB PDB	2ob4 PDB
1c4zAD PDB	2f4wAB PDB	2kjh PDB
1c4zAD PDB	2f4wAB PDB	1fbvC PDB

Figure 4.11 Partners of Template Interfaces page

Chapter 5

CONCLUSION

Detecting protein-protein interactions in the cell becomes much more important as large amount of protein structure data become available. Many of the processes in the cell are regulated by means of interactions among various proteins. There is a great increase in available experimental and computational methods to find interactions between proteins in recent years. Novel computational methods and computational tools have been developed and this is expected to do so in the near future.

In this work, we take advantage of an available prediction algorithm which uses both structure and sequence conservation in protein interfaces. We design a web accesible and database oriented software tool, namely PRISM web server, that is used to find interactions between target protein structures using this algorithm. This web server is used mainly to find whether there are interactions between input proteins or not. It is also used for examining the datasets employed in the algorithm: a set of protein interfaces extracted from protein complexes (templates) [8] and a list of proteins which are found to be similar to one or more of these templates in terms of structure and evolution [10]. These datasets and all related data is held in a relational database. Contents of this database is queried and displayed in a web interface. To explore the concepts behind the PRISM database in a visual environment, a graphical tool is integrated into PRISM's web interface.

In its current form, PRISM web server is a useful tool to predict interactions between two target proteins, among a list of proteins or a network of proteins. It also shows possible partners of protein structures in the template dataset. Although it offers useful features, much more can be added. Also, it has some shortcomings on performance and accuracy of the results.

As a future work, the set of templates and targets can be expanded to include new complexes and calculated predictions. Expanding the template set will allow us to find more possible interactions between input target proteins. Also, the web interface can be modified to provide possible different queries. For instance, users can choose the templates that they want to use when predicting interactions between proteins. By applying this particular restriction, the performance of the algorithm will be enhanced and predictions will be more purposive. With these improvements, we believe the next version of PRISM will be a useful tool to answer many other questions in the field of finding the protein-protein interactions.

In conclusion, predicting protein-protein interactions is a significant problem in systems biology. Even if there are different available methods to answer this question, there is no certain way to decipher all interaction networks in an organism. This question will continue to appeal scientific community to research on because such knowledge help researchers to identify nodes in pathways that cause disorders and design drugs that will impact on these nodes.

BIBLIOGRAPHY

1. Valencia, A. and F. Pazos, *Computational methods for the prediction of protein interactions.* Curr Opin Struct Biol, 2002. **12**(3): p. 368-73.
2. Ferrer, M. and S.C. Harrison, *Peptide ligands to human immunodeficiency virus type 1 gp120 identified from phage display libraries.* J Virol, 1999. **73**(7): p. 5795-802.
3. Gray, J.J., *High-resolution protein-protein docking.* Curr Opin Struct Biol, 2006. **16**(2): p. 183-93.
4. Halperin, I., et al., *Principles of docking: An overview of search algorithms and a guide to scoring functions.* Proteins, 2002. **47**(4): p. 409-43.
5. Andrusier, N., et al., *Principles of flexible protein-protein docking.* Proteins, 2008. **73**(2): p. 271-89.
6. Caffrey, D.R., et al., *Are protein-protein interfaces more conserved in sequence than the rest of the protein surface?* Protein Sci, 2004. **13**(1): p. 190-202.
7. Tsai, C.J., et al., *A dataset of protein-protein interfaces generated with a sequence-order-independent comparison technique.* J Mol Biol, 1996. **260**(4): p. 604-20.
8. Keskin, O., et al., *A new, structurally nonredundant, diverse data set of protein-protein interfaces and its implications.* Protein Sci, 2004. **13**(4): p. 1043-55.
9. Tsai, C.J., et al., *Protein-protein interfaces: architectures and interactions in protein-protein interfaces and in protein cores. Their similarities and differences.* Crit Rev Biochem Mol Biol, 1996. **31**(2): p. 127-52.
10. Aytuna, A.S., A. Gursoy, and O. Keskin, *Prediction of protein-protein interactions by combining structure and sequence conservation in protein interfaces.* Bioinformatics, 2005. **21**(12): p. 2850-5.

11. Keskin, O., et al., *Principles of protein-protein interactions: what are the preferred ways for proteins to interact?* Chem Rev, 2008. **108**(4): p. 1225-44.

12. Zhang, Q.C., et al., *PredUs: a web server for predicting protein interfaces using structural neighbors.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W283-7.

13. Guo, Y., et al., *PRED_PPI: a server for predicting protein-protein interactions based on sequence data with probability assignment.* BMC Res Notes, 2010. **3**: p. 145.

14. Singh, R., et al., *Struct2Net: a web service to predict protein-protein interactions using a structure-based approach.* Nucleic Acids Res, 2010. **38**(Web Server issue): p. W508-15.

15. Fox, N., et al., *KINARI-Web: a server for protein rigidity analysis.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W177-83.

16. Nitsch, D., et al., *PINTA: a web server for network-based gene prioritization from expression data.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W334-8.

17. London, N., et al., *Rosetta FlexPepDock web server--high resolution modeling of peptide-protein interactions.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W249-53.

18. Lopez, G., A. Valencia, and M.L. Tress, *firestar--prediction of functionally important residues using structural templates and alignment reliability.* Nucleic Acids Res, 2007. **35**(Web Server issue): p. W573-7.

19. Lopez, G., et al., *firestar--advances in the prediction of functionally important residues.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W235-41.

20. Lopez, G., A. Valencia, and M. Tress, *FireDB--a database of functionally important residues from proteins of known structure.* Nucleic Acids Res, 2007. **35**(Database issue): p. D219-23.

21. Soding, J., *Protein homology detection by HMM-HMM comparison.* Bioinformatics, 2005. **21**(7): p. 951-60.

22. Altschul, S.F., et al., *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* Nucleic Acids Res, 1997. **25**(17): p. 3389-402.

23. Tress, M.L., O. Grana, and A. Valencia, *SQUARE--determining reliable regions in sequence alignments.* Bioinformatics, 2004. **20**(6): p. 974-5.

24. Mitra, P. and D. Pal, *PRUNE and PROBE--two modular web services for protein-protein docking.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W229-34.

25. *Jmol: an open-source Java viewer for chemical structures in 3D*. 2004-2013; Available from: http://www.jmol.org/.

26. Walsh, I., et al., *CSpritz: accurate prediction of protein disorder segments with annotation for homology, secondary structure and linear motifs.* Nucleic Acids Res, 2011. **39**(Web Server issue): p. W190-6.

27. Warde-Farley, D., et al., *The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function.* Nucleic Acids Res, 2010. **38**(Web Server issue): p. W214-20.

28. Ogmen, U., et al., *PRISM: protein interactions by structural matching.* Nucleic Acids Res, 2005. **33**(Web Server issue): p. W331-6.

29. Ogmen, U., *PRISM : A System for the Query, Visualization and Analysis of Protein Interfaces and Their Networks*, in *Graduate School of Engineering*. 2006, Koc University.

30. Eckerson, W.W., *Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications.* Open Information Systems, 1995. **10**.

31. Howitz, C., *What Is 3-Tier(Multi-Tier) Architecture And Why Do You Need It?*, in *SimCrest*. 2012.

32. *RabbitMQ Messaging that just works*. 2010-2013; Available from: http://www.rabbitmq.com/.

33. O'Hara, J. *Toward a commodity enterprise middleware*. Acm Queue, 2007. **5**, 48-55.

34. Mashiach, E., R. Nussinov, and H.J. Wolfson, *FiberDock: Flexible induced-fit backbone refinement in molecular docking.* Proteins, 2010. **78**(6): p. 1503-19.

35. Aytuna, A.S., *A High Performance Algorithm For Automated Prediction of Protein-Protein Interactions*, in *Graduate School of Engineering*. 2004, Koc University.

36. Nussinov, R. and H.J. Wolfson, *Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques.* Proc Natl Acad Sci U S A, 1991. **88**(23): p. 10495-9.

37. Shatsky, M., R. Nussinov, and H.J. Wolfson, *A method for simultaneous alignment of multiple protein structures.* Proteins, 2004. **56**(1): p. 143-56.

38. Tuncbag, N., et al., *Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using PRISM.* Nat Protoc, 2011. **6**(9): p. 1341-54.

39. Hubbard, S.J.T., J.M., *'NACCESS', Computer Program*. 1993: Department of Biochemistry and Molecular Biology, University College London.

40. Mashiach, E., R. Nussinov, and H.J. Wolfson, *FiberDock: a web server for flexible induced-fit backbone refinement in molecular docking.* Nucleic Acids Res, 2010. **38**(Web Server issue): p. W457-61.

41. *World Wide Web (WWW).* 1990-2013.

42. *HTML (HyperText Markup Language)*. 1991-2013; Available from: http://www.w3.org/html/.

43. *PHP: Hypertext Preprocessor*. 1995-2013; Available from: http://www.php.net/.

44. *MySQL: The world's most popular open source database*. 1995-2013; Available from: http://www.mysql.com/.

45. *Phyton Programming Language*. 1991-2013; Available from: http://www.python.org/.

46. Herraez, A., *Biomolecules in the computer: Jmol to the rescue.* Biochem Mol Biol Educ, 2006. **34**(4): p. 255-61.