LISA: A Fast Filtering Algorithm for Structural Alignment

by

Emre Küçük

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Data Science

KOÇ ÜNİVERSİTESİ

August 3, 2022

LISA: A Fast Filtering Algorithm for Structural Alignment

Koç University

Graduate School of Sciences and Engineering

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

Emre Küçük

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. Dr. Attila Gürsoy (Advisor)

_____

Assoc. Prof. Dr. Didem Unat

_____

Asst. Prof. Dr. Sefer Baday

Date: _____

Dedication, devotion. Turning all the night time into the day.

ABSTRACT

LISA: A Fast Filtering Algorithm for Structural Alignment
Emre Küçük
Master of Science in Data Science
August 3, 2022

Protein - Protein Interactions (PPI) cause vital processes such as growing, division or maintenance of a cell. It is an important topic in order to understand the functioning of any cell with itself and others. Even though the most reliable techniques are experimental for investigating the properties of different structures, numerical methods are much faster with negligible error. Nevertheless, predictions of PPI needs improvements in order to derive more accurate results faster. In this thesis, we are implementing a two-step hashing algorithm in order to increase the speed of prediction of PPI structures. In the first part, a large number of interfaces are classified by their different properties such as bond angles, dihedral angles and distances between Carbon Alpha (CA) atoms. For each non-consecutive residue that has the distance of 4 $\mathring{A}$ to 13 $\mathring{A}$, between CA atoms,we are calculating necessary angles and distances with selected CA atoms and their consecutive neighbors. Then, we are classifying fragments of interfaces with similar properties in the first phase of the algorithm by using a hash table. In the second phase, we are comparing a given protein using the same properties that calculated for templates, and score them by their similarity. Proposed algorithm is developed for filtering the dissimilar interfaces and limiting the possible number of interfaces that can be used in Template-Based PPI prediction protocols. In addition, it is useful for reducing the computation time of any structural alignment algorithm to find input templates for a docking algorithm by returning a filtered subset from a given template dataset.

ÖZETÇE

LISA: Yapisal Hizalamanın Hızlanması için Hızlı Bir Filtreleme Algoritmasi

Emre Küçük

Veri Bilimleri, Yüksek Lisans

3 Ağustos 2022

Protein - Protein Etkileşimleri (PPE), bir hücrenin büyümesi, bölünmesi veya bakımı gibi hayati süreçlere neden olmaktadır. Herhangi bir hücrenin kendisi ve diğerleri ile işleyişini anlamak için önemli bir konudur. Farklı yapıların özelliklerini araştırmak için en güvenilir teknikler deneysel metotlar olsa da, sayısal yöntemler ihmal edilebilir hata ile çok daha hızlıdır.Bununla birlikte, daha doğru sonuçların daha hızlı elde edilmesi için PPE tahminlerinin iyileştirilmesi gerekmektedir. Bu tezde, PPE yapılarının tahmin hızını artırmak için iki aşamalı bir karma fonksiyonu algoritması sunulmaktadır.Birinci bölümde, çok sayıda arayüz; bağ açıları, dihedral açılar ve Karbon Alfa (CA) atomları arasındaki mesafeler gibi farklı özelliklerine göre sınıflandırılmaktadır. Seçilmiş iki CA atomunun 4 $\mathring{A}$ ile 13 $\mathring{A}$ arasındaki mesafeye sahip ardışık olmayan her amino asit için, bu CA atomları ve onların komşu CA atomları ile gerekli açıları ve mesafeleri hesaplanır. Daha sonra, bir hash tablosu kullanarak algoritmanın ilk aşamasında benzer özelliklere sahip amino asit dörtlüleri özelliklerine ve arayüzlerine göre sınıflandırılır. İkinci aşamada, arayüzler için hesaplanan aynı özellikler kullanılarak belirli bir protein ile karşılaştırılır ve benzerliklerine göre puanlanır. Bu algoritma, Arayüz-Bazlı PPE tahminleme algoritmalarında seçilecek arayüzler arasından alakasız seçenekleri filtrelemek ve olası adayları sınırlandırmak için geliştirilmiştir. Protein yerleştirme algoritmalarında girdi olarak kullanılacak arayüzleri bir veri setinden seçmek için kullanılacak yapısal hizalama algoritmasının çalışma süresini veri setini filtreleyip küçülterek azaltmak için kullanışlıdır.

ACKNOWLEDGMENTS

First of all, I would like to begin with thanking my advisors Prof. Dr. Attila Gürsoy and Prof. Dr. Özlem Keskin Özkaya. The chance they give me to work in COSBILAB and becoming a COSBIAN will be something I will carry with me proudly through my entire life. I feel that their wisdom, vision and mentoring made me a better person.

I appreciate my thesis committee members Assoc. Prof. Dr. Didem Unat and Asst. Prof. Dr. Sefer Baday , for their time and guidance. I am grateful for their opinions and critiques about my thesis.

I also would like to thank all my friends from COSBILAB, we built very strong friendships that will last forever. Thank you for helping me get through this process, both emotionally and technically. I can not thank enough for your support. Kayra Kösoğlu, Zeynep Abalı and Simge Şenyüz, I individually thank you for everything. We have been through thick and thin together, and your friendship is invaluable to me.

I owe the deepest gratitude to Oğuz Öztınaz, Remzi Osman Gönültaş and Anıl Ulusoy, I would not make it to my undergrad years without your support. I am immensely jubilant to be your friends. I hope our friendship lasts a lifetime.

Finally, I would like to thank to my family. You always believed in me and never doubted on me. You have always respected my own decisions and supported me. Thank you for pushing me to my highest potential. I will keep improving myself and make you proud.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

PPI	Protein-Protein Interaction
PRISM	Protein Interactions by Structural Matching
PDB	Protein Data Bank
LISA	Little's Structural Aligner

Chapter 1

INTRODUCTION

Protein-Protein Interactions (PPI) control vital processes such as growth, division, or maintenance of a cell. It is an important topic to understand the functioning of any cell with itself and others. Even though the most reliable techniques are experimental for investigating the properties of different structures, numerical methods are much faster with negligible error. Nevertheless, the prediction of PPIs requires improvements to derive more accurate results faster. In this project, we are implementing a heuristic algorithm for structural matching in a fast manner. The main motivation is to calculate similarity scores and filter templates from a given template database in order to further use in Protein-Protein interaction prediction for a given arbitrary structure. Doing that in a fast way provides docking methods to work far faster by negligible error.

In this project, we developed a new filtering algorithm called LISA. LISA is based on and inspired by a docking algorithm called SnapDock (Estrin and Wolfson, 2017). In addition, this thesis is a continuation of QuickRet (Balcı, 2018), with improvements in the algorithm.

Total number of structures in the Protein Data Bank (Berman et al., 2000) are 188431 by 2022 and the increase is exponential. Having a large number of biological data emerges new possibilities of using computational methods for advancing the understanding of nature. Moreover, having an exponential increasing in the data comes with its own problems; such as storage and time overheads. More data can be used to obtain more accurate results, but the time required to process larger data sets might create a burden. LISA aims to tackle the time dimension of the computational overhead problem.

LISA is a filtering algorithm developed to limit the number of possible structural alignments between a given data set of interfaces and an arbitrary structure. Currently, PIFACE (Cukuroglu et al., 2014) has 22604 different interface clusters, which means for a non-redundant interface data set, an arbitrary structure have 45208 different options that can be aligned. In addition, most of the interfaces will not be similar, hence dissimilar options are not relevant. They neither will be used to understand the structure nor be used as template for template based docking methods. Thus, having a filtering algorithm is a good practice for limiting the number of possible similar options that can be aligned. LISA aims to put an upper limit to the number of possible similar interface options that any arbitrary structure can have.

Chapter 2

LITERATURE SURVEY

2.1 Proteins

Proteins are vital for the functioning of the cell. Interaction between two or more protein structures cause different operations to begin. They are consist of amino acids. Structure of an amino acid can bee seen in figure 2.1.

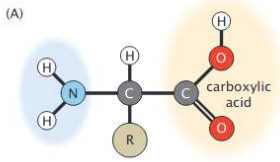


Figure 2.1: Structure of an amino acid (Bahar et al., 2017)

An amino acid have a central Carbon atom referred as Carbon Alpha (C_α). There are four different parts bonding with C_α, an amino group (NH_2), a carboxyl group ($COOH$), a Hydrogen atom and side chain group (R). Every amino acid has the same structure. Only difference between them is the side chain group. There are twenty different amino acids. Amino acids come together and form chains. One or several chains conform a protein structure.

These amino acids conform a sequence. This sequence is called "primary structure". Three dimensional folded form of the protein is called as "tertiary structure", including "secondary structure" elements such as α-helix and β-sheet which are folded in a certain way. Secondary structure elements can be seen frequently in a majority of the structures as a pattern. These structures conform by different bonding of the side chains and other atoms. Structures with more than one chain has "quarternary structure" form.

Proteins cooperate or interact with each other for a goal, hence it is probable to understand the functions of a protein by understanding its binding participants (Deng et al., 2002). Hence, it is important to understand the interactions between different protein structures.

Protein structures need several experimental techniques to be determined. X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy, and cryoelectron microscopy can be used to acquire structures of proteins (Lu et al., 2010). Experimentally extracted structures are stored in a database called Protein Data Bank (PDB) (Berman et al., 2000). Number of structures in PDB grows exponentially. Total number of structures and annually determined structures year by year between 1972 - 2015 can be seen in figure 2.2.

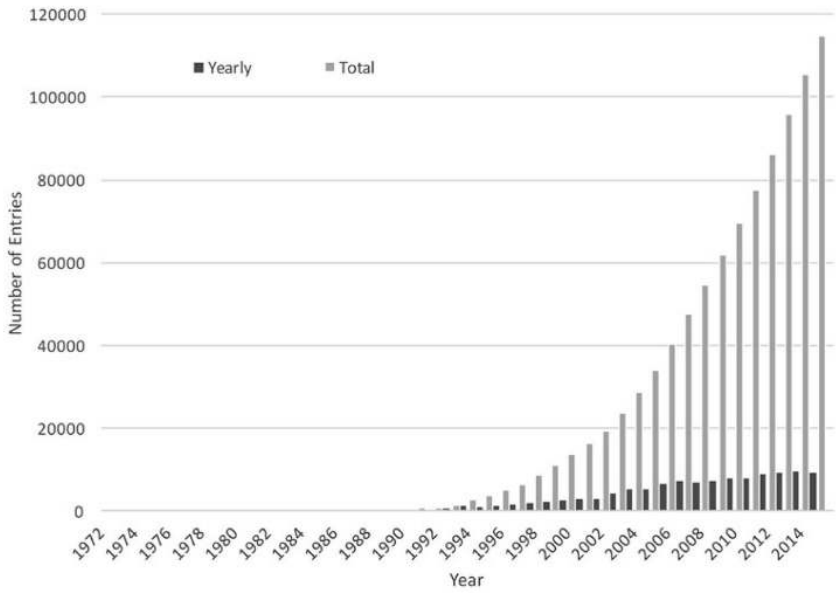


Figure 2.2: Number of structures in PDB (Burley et al., 2017)

2.2 Related Work

Cellular functions in the cell such as metabolism, information processing, decision making, transport, and structural organization are mainly operated around proteins

(Braun and Gingras, 2012). These processes mainly controlled with Protein - Protein interactions. Hence, Protein-Protein interaction prediction is a highly interesting area in bioinformatics. Even though there are various techniques for predicting a PPI, template-based methods are determined to be the most accurate one than other techniques such as free docking (Porter et al., 2019). There are different tools available for Protein-Protein interaction prediction on the internet (Rao et al., 2014; Szilagyi and Zhang, 2014). PRISM (Baspinar et al., 2014; Tuncbag et al., 2011) is one of the crucial protocols for predicting PPIs. For given two targets and a template, PRISM predicts an interaction between the targets using a selected template as a reference. With this work, we are aiming to filter out the majority of the non-similar templates from consideration for using in PPI prediction for target structures. PRISM consists of four successive steps:

- Surface extraction
- Structural alignment and hot spot analysis
- Transforming target structures into template interfaces and filtering
- Docking

Template-based algorithms work with docking. Assuming we have two structures that might interact, we can try using experimentally discovered structures as templates, meaning that we can reference their way of interaction to the target structures' interactions we try to find.

With that said, two questions immediately occur: what are we going to use as a reference? The second question is how are we going to use it? For the first question, we used PIFACE (Cukuroglu et al., 2014), a non-redundant protein interface database. PIFACE was designed to be used as a database of template structures for Protein-Protein Interaction Prediction. For the second question, even though there are different docking algorithms in the literature (Grosdidier et al., 2011; Estrin and Wolfson, 2017; Mashiach et al., 2010), finding a valid template is a difficult task and requires computational power and time. To obtain the best possible template,

we can make structural alignment between two target structures (Shatsky et al., 2002) and upon their similarity, we can decide if a possible interface structure is a valid template for our problem or not. Considering we have a list of non-redundant interface structures of PDB coming from PIFACE (Cukuroglu et al., 2014), we could search the entire database and find the best matching interface as a template.

But searching entire PDB (Berman et al., 2000) or even its non-redundant version PIFACE (Cukuroglu et al., 2014) would take drastically long time, considering the number of structures. Hence, we need a fast structural alignment algorithm for searching through a database and find the best matches amongst them. Entering LISA; a structural alignment algorithm that uses the benefits of hashing and finding best matches in a fast manner.

There are different structural alignments can be found in the literature. Geometric hashing based ones are showing similarity as they use a very similar technique to LISA (Nussinov and Wolfson, 1991; Fischer et al., 1992). We used Multiprot (Shatsky et al., 2002) and TM-Align (Zhang and Skolnick, 2005) in our studies to compare LISA's filtered results. There are other alignment algorithms emphasized around efficiency as well. MADOKA (Deng et al., 2019) is one of the algorithms that is focusing on efficiency. It uses secondary structures in the alignment process. LISA on the other hand, does not rely on secondary structures and can be used on C_α only structures as well. Methods behind LISA allows using any kind of point cloud sets to be filtered. Another advantage of LISA is that its filtering technique is sequence independent. Smolign (Sun et al., 2011) is another structural alignment algorithm that uses predefined motifs and RMSD as a quality of alignment. LISA has its own unique scoring function. Using RMSD as a performance metric is not the best option for some cases according to Zhang and Skolnick (Zhang and Skolnick, 2004).

2.3 Ground Work

LISA is a data-driven algorithm that is built with a hash table. One of the crucial aspects of hashing is obtaining stored information very fast to recognize different

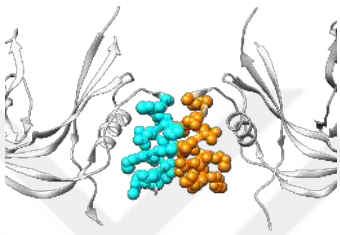
objects. Information is stored in a hash table so that it would be possible to obtain when it is necessary. Instead of searching through a database, we can use predetermined features and elicit desired information with feature references (Wolfson and Rigoutsos, 1997). We used a similar approach in LISA by calculating several features and storing them inside a hash table.

Second inspiration of LISA algorithm is Pose Clustering method (Stockman, 1987). After storing information in a hash table, LISA retrieves similar objects from perceived features of a given structure. Then, using Kabsch algorithm (Kabsch, 1978), LISA finds parameters to align the stored information and observed scene (given query structure). Moreover, clustering the parameters of Kabsch algorithm generates a strong information about similarities between stored information and observed object.

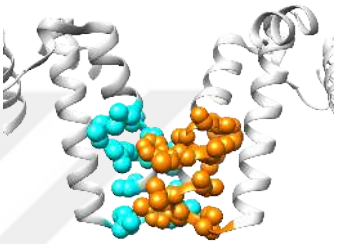
2.4 Data

To construct the hash table, we used a dataset of interfaces obtained from PIFACE (Cukuroglu et al., 2014). In PIFACE, protein interfaces are defined by contacting and nearby residues. Contacting and nearby residues are determined by HotPoint (Tuncbag et al., 2009). After determining interfaces, they are grouped by using betweenness (Freeman, 1977) and creating a graph network according to their similarity. Then interfaces with a betweenness score higher than a threshold are clustered as they are similar. In the end, a non-redundant database is created with groups of analogous interface structures. For each cluster, there is a cluster representative to represent the cluster structure and its members. LISA's lookup table has created by obtaining interfaces of cluster representatives. Their information is kept as contacting and nearby residues. As a result, the dataset contains 22604 interfaces, matching the number of clusters and representing the entire PDB database until 2013 in a non-redundant manner.

Within a list of 22604 interfaces, we consider each interface as a possible different structural matching candidate. Interfaces have two chains which treated individually. An arbitrary structure can be similar to either side of an interface. Hence,



2O3O,AB



1YXB,BD

2VLV,AB

Figure 2.3: Example interfaces from the dataset. First four characters represent the PDB id of a structure, last two characters represent the chain id's of an interface. In the figures, only the chains that interfaces belong to are shown. Orange and cyan colored parts represent contacting and nearby residues, while white parts represent the rest of the chains.

we must align for each chain rather than matching the entire interface with a given structure. To find a structural similarity for an arbitrary complex within a list of interfaces fast, we will determine possible candidates for protein-protein interaction templates to use in docking methods.

In the experiments, we also used database of PRISM's results by picking random targets and comparing their templates with LISA's results.

Chapter 3

METHOD

In this section, we are going to explain the method that we develop in order to achieve fast, accurate structural alignment for given structures. As a reference, we are using the PPI template database PIFACE. After extracting small fragments from entire template representatives, we are making a pre-processing stage and storing every fragment from every template in a hash table grouped with determined features. Then, for an arbitrary structure, we are finding small fragments and extracting the same features. We are matching fragments with same feature set, divided into groups by template ids.

Algorithm has two stages. In the first stage, we are searching through a list of interfaces and storing them into a hash table. After this process, LISA requests a structure to compare with the hash table. Then we look their similarities in the second stage.

3.1 Preparation Stage

In the beginning, we have 45208 interface chains to search through that contains C_α atoms. Other atoms are discarded.

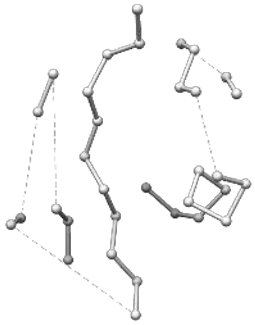


Figure 3.1: CA atoms of 2VLV,AB interface. Light gray balls are representation of CA atoms.

To complete the hash table preparation, we are extracting small parts from each interface side, called "fragments". Fragments are small pieces obtained from the interface structure that contains C_α atoms of two residues within the C_α distance of 4 Å and 13 Å and their consecutive neighbors of each.

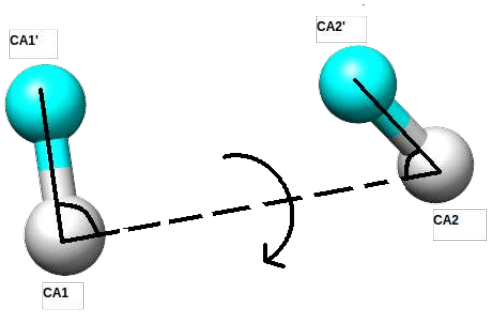


Figure 3.2: Example fragment. Extracted features will be used as key for the hash table, whereas values will be coordinates and interface chain ids.

Interfaces consist of structurally close residues. Generally, they appear sequentially close residues from different parts of the chain. As a result, for each C_α atom in an interface, we filtered other C_α atoms within the given distance. If two C_α atoms of different residues are within the range of 4 Å and 13 Å, we build fragments using four atoms: two are selected C_α atoms and other two are their consecutive C_α atoms. By this idea, we are aiming to have an unambiguous representations of interfaces while maintaining sequence independent status.

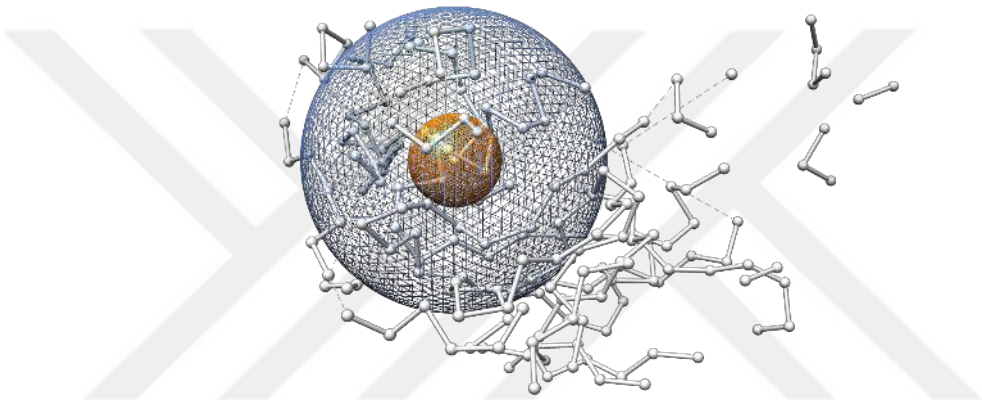


Figure 3.3: Part of 2XKX,AB interface. In this example, residue 123 ARG is used. Each CA pair which are between blue and orange spheres and their consecutive C_α atoms will be used for creating fragments. All CA atoms will be measured with each other and ones with $4 < d_{C_\alpha} < 13$ will be used for building a fragment.

The process of creating fragments will be done for each contact and nearby residue in the interface. The algorithm iterates through each residue and will search their neighbor fragment candidates by iterating other residues and checking their d_{C_α}. Hence, the creation of fragments will create a time complexity of $O(n^2)$. Upon creation of a fragment, its information will be stored in the hash table.

Hash tables consist of key-value pairs. In LISA, keys are four different integers created by using atoms in fragments. Values are tuples; the first value of the tuple is the coordinates of atoms, the second value of the tuple is the interface chain id.

For simplicity sake, let's call first and second residue's Carbon Alpha atoms as

$C_\alpha^1, C_\alpha^{1'}, C_\alpha^2, C_\alpha^{2'}$. For each fragment, four features will be extracted as hash table keys:

- Distance between C_α^1 and C_α^2.
- Bond angle between $C_\alpha^{1'}, C_\alpha^1, C_\alpha^2$.
- Bond angle between $C_\alpha^1, C_\alpha^2, C_\alpha^{2'}$.
- Dihedral angle by all atoms.

After calculating features for a fragment, it will be inserted to a hash table. Keys are calculated features and values are tuples with fragment atom coordinates and interface chain ids.

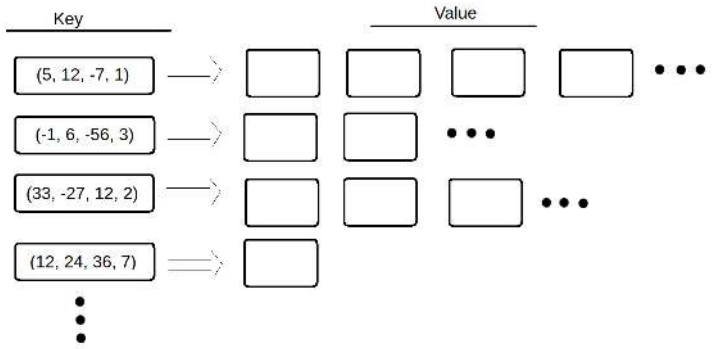


Figure 3.4: Hash table representation. There might be different fragments with the same feature set, in which they will be grouped in the same list under the identical set of keys in the hash table.

After inserting every fragment belonging to 45208 interface chains, the hash table is ready to use. The algorithm for creating the hash table is in Algorithm 1.

To obtain some possible matches with a reasonable amount, we are discretizing our features with a set of threshold values. Otherwise, the possibility of obtaining a match would be minimal. Discretization is handled by making an integer division

Algorithm 1 LISA - Stage 1

```
1: Data: List of interface structures
2: Result: Hash table of features extracted from the interface structures
3: initialize hashtable
4: for interface in the interface list do
5:     read interface from PDB file
6:     assign a model number i to the interface
7:     list of feature-fragment pairs ← extractFeatures(interface)
8:     for f, b in list of feature-fragment pairs do
9:         key ← discretize f
10:        insert (b, i) into hashtable(key)
```

within all features with predefined thresholds. These values are defined as 5 for dihedral angle and bond angles, 1 for the distance between Carbon Alpha atoms. Using determined thresholds will enable close angles to be binned in the same hash table key.

For example, if one fragment has the set of features as $(-127.82, 16.33, 45.00, 12.2)$, assuming the order is the dihedral angle, first bond angle, second bond angle, and distance between Carbon alpha atoms, the key extracted for the base is $(-25, 3, 8, 12)$. Any other fragment that has the same set of key integers will be grouped. For a calculated feature v and threshold t, all values within the range v and $v + t$ will put into the same bin as $[\lfloor v/t \rfloor, \lfloor (v + t)/t \rfloor)$. Thresholds are determined experimentally when we increased the number as 10 for each angle and 1 for the distance of Carbon alpha atoms, the length of the lists in the hash table increased vastly and created a computational time overhead.

3.2 Alignment Stage

3.2.1 Fragment Matches

After the creation of the hash table, the preparation stage is finished. The hash table is ready to use after this point. The algorithm expects a structure to compare

and make alignments by using the hash table. Let's call an arbitrary structure where we want to find possible good alignments as "query structure". The first step is to extract all fragments from the query structure. For each fragment, the algorithm will calculate the same features defined in the hash table building stage. If there is a match between any key and the calculated features, all of the matches in the list will be retrieved. Hence, with just one calculation and lookup, every similar fragment matches amongst interface chains will be fetched. For each match, we will group matches into a new hash table according to their interface chain ids. New list will be called the match list. Keys will be the interface chain IDs, whereas values will be tuples. The first item of the tuple will be the coordinates of the query structure's current fragment. The second item of the tuple will be the coordinate of the retrieved match's fragment that belongs to the interface.

Furthermore, when we obtain a match from the hash table, it consists of a list of tuples. The list of tuples contains all elements with the same features, belonging to various interfaces. Each tuple has two elements. The first element is the coordinate of the matched fragment, the second element is the interface chain id. The algorithm uses the interface chain id as the key for the match list. Then it creates a new tuple by using the coordinate of the current iteration's fragment and "matched fragment"s coordinates. Next inserts it with key-value pair to the match list. This operation happens for each element of the list for each repetition. In every iteration, there will be a new list. Previous lists might occur again, but this time we are appending new tuples with different fragments.

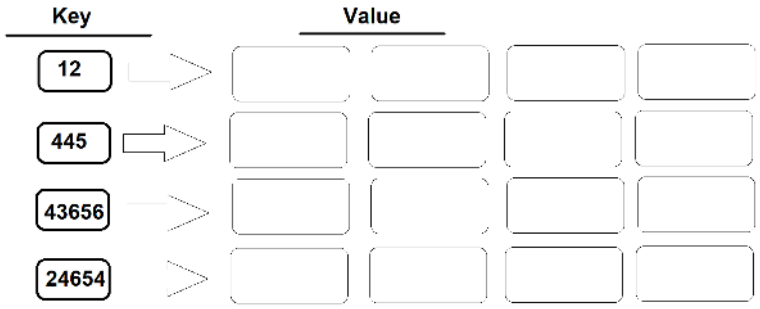


Figure 3.5: Match list representation. We can obtain the entire matches within each interface individually after full insertion.

In the process of insertion to the match list, we will create new keys every time we come across a new interface chain. Otherwise, we will append new tuples to the existing lists in the match list. Intuitively, this list will contain every possible match from query structure fragments with fragments that come from the database, divided by their interface chain ids.

This two-step process increases the lookup speed for the entire structure drastically. With one fetch, we can get all fragment matches from all interface chains. One small caveat is we do not have the relative spatial information between fragments. But this challenge will be tackled by the usage of transformations.

3.2.2 Transformations

As stated above, fragments do not contain relative spatial information. Even though fragments of an interface chain and query structure might match, their positions might be entirely different hence this does not give information about the complete structure of the interface. Matched fragments of the query structure might belong to irrelevant parts, even they do not have to be on the surface. Transformation is used as a solution to tackle this challenge.

After obtaining the match list, the algorithm iterates through every interface, which means every possible tuple in it. In the iteration process, for each tuple, calculates structural alignment parameters using Kabsch algorithm (Kabsch, 1978).

Kabsch algorithm returns three parameters, two vectors for transforming each component's center of mass to the origin, and a matrix for rotation. By rotation matrix, the algorithm extracts Euler angles to rotate one fragment onto the other.

Euler angles are the key parameters to a third hash table for storing transformations. Keys are also discretized. This time hash tables are individually calculated for each interface for using instantly to find the parameter with the most fragments in it. Searching through transformation hash tables, the algorithm finds the longest transformation. Intuitively, the longest transformation results in a simple deduction: if an interface has a long list of fragment transformations with the same parameters, fragments that match might be integral in query structure. Because we are shifting and transforming using the same parameter set for numerous fragments.

So we can infer that if a query structure and interface chain have a long list of fragment transformations with the same parameters, they might belong to an interface.

3.2.3 A Novel Idea to Tackle Local Transformation Problem

Local transformations do not reflect global transformations. Thus, two different problems are occurring using the Kabsch algorithm in the hashing approach. The first problem is that the Kabsch algorithm uses the center of mass of two sets of points to reduce the RMSD between them. However, when local transformations are applied, their parameter set might not be similar and emulate the global transformation. The second problem is that even though query structure and an interface are similar, the interface is likely to be a smaller set of points compared to the query structure. As a result of this, their center of masses might be different. Even using the Kahsch algorithm between the complete structure and complete interface without hashing them might not work, since essentially, Kabsch is not an alignment algorithm. It is originally an algorithm for reducing RMSD between two "point clouds."

We are going to explain the problems in a simple example. 1hx1 and 3ldq belong to same cluster in PIFACE. 3ldq is also the representative and 1hx1 is one of the

members. Both of the structures have the same B chain, their FASTA sequences are the same. However, they are aligned differently in the coordinate system. In figure 3.6, we can clearly see that if we take two different fragments and align them, their translation vectors will not be the same. If we select a base from one side of the interface from both structures and align them, its translation vector will not be the same as bases that we select from both structures that belong to other side of the interface. Translation vectors are different than each other, which can be seen in figure 3.6. In most cases, rotation vector will be different as well.

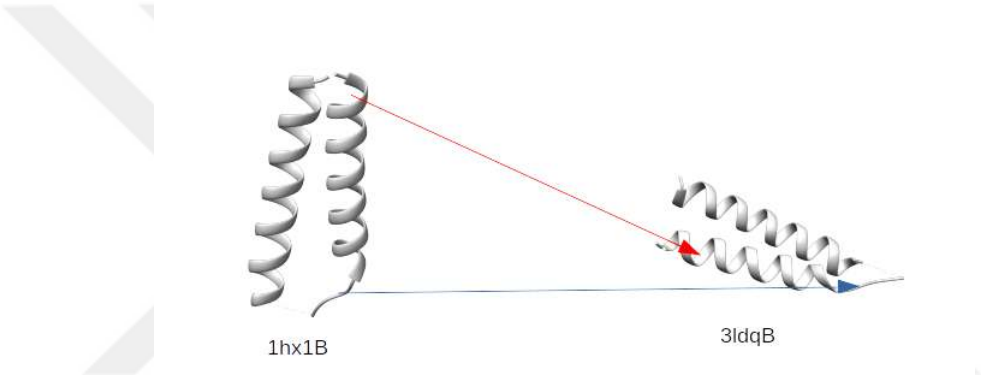


Figure 3.6: Example local translation vectors from selected different bases of 1hx1's B chain and 3ldq's B chain.

In second problem, center of masses need to be the same for the parts that should be aligned. Entire structure probably have a different center of mass from the similar part. Center of mass of the similar part of the query structure will be close to the surface, and if we are going to apply the Kabsch algorithm; somehow we must find a appropriate center of mass that considers the correct region. An example can be seen in the figure 3.7.

Figure 3.7: 2o3o interface's A chain is on the left, and entire chain of 1ylf structure's A chain on the right. Magenta parts are PIFACE's clustered residues, hence they need to be aligned. Clearly if we apply the Kabsch algorithm it will try to align them from their center of masses. But we need to set the center of mass of 1ylf to the magenta parts.

We came up with a solution to attack these two problems. Instead of locally transforming each match-list pair, we combined every tuple and aligned them. In the final, we have $C(n, 2)$ elements of match-list tuple duples. Moreover, when we take combinations of the entire match list, we are shifting the center of mass between these duples. Hence, center of mass will be set to correct position in some cases. Their alignments and features are going to be very similar, hence they will be clustered on the same transformation table. This will form the longest transformation. For an arbitrary structure and interface, simple examples can be found in figures 3.8, 3.9, 3.10.

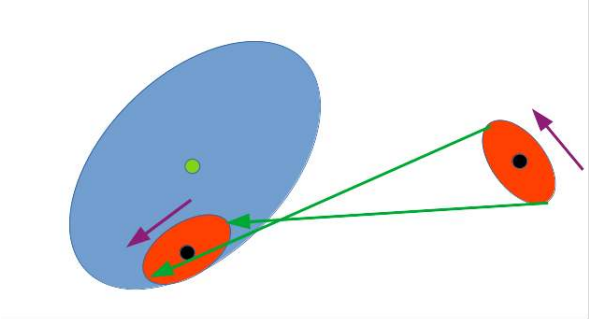


Figure 3.8: If we transform singular fragment pairs, their parameters will be different. In this example, an interface on the right is hashed and aligned to an arbitrary structure on the left. Assume blue is entire structure, orange parts are similar parts and their similarity is in the direction of purple arrow. If we choose a base from the side of beginning of the arrow, we will transform it to beginning part of the structure. Same logic applies when transformation happens at ending side of the purple arrow. Thus their translation vectors will be different, even though all fragments belong to the similar regions. Translation vectors are represented as green arrows.

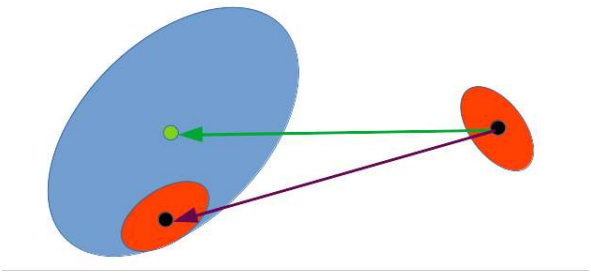


Figure 3.9: An illustration of the second problem. If we try to align without hashing, the Kabsch algorithm will align the center of masses and rotate one structure onto the other. Then center of mass of the interface (structure on the right) will be aligned to center of mass of structure of the left (green dot), though it should be translated to center of mass of the similar region. Translation should be on the direction of purple arrow instead of green arrow.

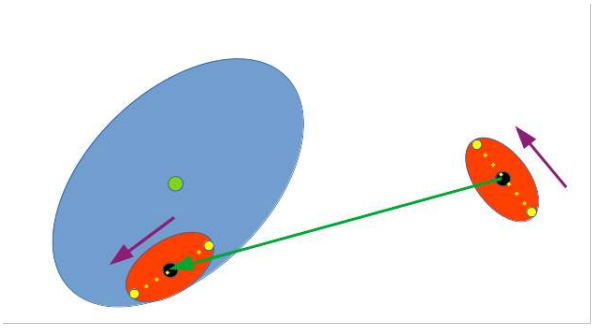


Figure 3.10: Solution to both problems. When we choose the similar fragment pairs as duples, some of the duples will have the center of mass of similar region. Then the transformation will occur in correct way. Yellow dots represent center of masses of exemplary fragments and their center of mass will be desired correct answer. All of the fragment duples which have the center of mass of similar region will contain the correct transformation parameter, and if there is an actual similarity, number of fragment duples with determined parameter set will have a high number of elements. In addition, using hashing and fragments by nature in LISA automatically solves the second problem. Instead of transforming entire structures from center of mass to center of mass of entire structure and entire interface, it transforms from center of mass of a fragment to another one. This condition automatically solves the second problem.

3.2.4 Scoring

In LISA, we used two partial scores and added them up. The first score comes from the length of the match list. The second score comes from the value of the longest transformation list with the same parameters. There are two key points to consider here: interface lengths are different, and transformation information is rather more important; because gives inference about relative spatial properties.

For the first score, we looked at the lengths of matches in the match list. For an arbitrary interface, the length of the list is considered the first score. However, there are interfaces with different lengths. To normalize this score, we divided the length of the match list with the number of fragments coming from the interface.

For the second score, we considered the length of the longest transformation. To normalize this score, we divided the transformation match-list length with the related interface's length of the match list.

$$\frac{length(matchlist[i])}{sum(bases_i)} + \frac{(length(max(bestpose_i)))^2}{length(matchlist[i])}$$

This concludes the alignment stage. In the end, the algorithm produces similarity scores for each interface. After the sorting, it is possible to obtain the determined number of best matches. This property enables the algorithm to tweak the sensitivity of results. It is possible to have a "top n" approach and select several interface matches amongst the entire data set. Second stage of LISA can be seen in Algorithm 2.

3.3 Main Idea

By going through each side of each interface, we are creating a table that consists of small parts of each interface side. When we try to align an arbitrary structure and find structural matches, we investigate small chunks amongst all structures. To achieve this goal, we need to create fragments for the query structure same as our lookup hash table. After this operation, we can extract the same features and check for similarities by using our lookup table.

Why this work? Mainly, the hash table's keys are spatial properties, and for a match with arbitrary structure and hash table, we can extract the fragments with the same spatial properties from the entire data set with just one lookup. The only concern is their relative spatial positions on alignment. We have the necessary information about spatial equities about fragments, but we do not have relative positions between each other. This might occur as a problem in the future since we do not store the relative information in the lookup table for alignment. This problem is tackled by using the Kabsch algorithm and combining every element of match-list as $C(n, 2)$. Fragments with the same transformation parameters as an interface might form the shape of the interface structurally. In final, we are looking for the lengths of these matches and calculate similarity for every interface. Then

Algorithm 2 LISA - Stage 2

Data: Query Structure
Result: Similarity score for each interface
read protein structure from PDB file
list of feature-fragment pairs ← **extractFeatures**(protein)
initialize $matchlist$
for f, b_q in list of $feature - fragment$ pairs **do**
    $key \leftarrow$ discretize f
    retrieve $(b_{inteface}, i)$ pairlist from $hashtable(key)$
    for $(b_{interface}, i)$ in pairlist **do**
        insert $(b_{interface}, b_q)$ into $matchlist[i]$
initialize scoretable
$score_1 \leftarrow$ **calcScore**(matchlist)
for i in $matchlist$ **do**
    initialize posetable
    for $(b_{interface}, b_q)$ pair in $C(matchlist[i], 2)$ **do**
        $transformation \leftarrow$ **calcTransformation**$(b_{interface}, b_q)$ between pairs
        key ← discretize $transformation$
        insert transformation into posetable(key)
    bestpose ← **length**(**max**(posetable))
    $score_2 \leftarrow$ **calcScore**(bestpose, **length**(matchlist[i]))
    scoretable[i] ← $score_1 + score_2$
filteredList ← Sort results and discard matches below the given threshold

we can sort the results and filter the wrong results by a given hyper parameter threshold.

3.4 Software

LISA is purely developed by Julia Programming Language version 1.7.2 (Bezanson et al., 2017). To parse PDB files and apply transformations between fragments BioStructures.jl package has been used (Greener et al., 2020). Figures are created and investigated by UCSF Chimera (Pettersen et al., 2004).

Chapter 4

EXPERIMENTS

4.1 PIFACE Experiments

We conducted several tests using PIFACE data set. Finding similarities between structures is a difficult task, considering there are 22604 clusters of interfaces. A structure can be similar to each side of a representative, thus it gives 45208 possible candidate solutions. In PIFACE, these clusters have members of Protein - Protein interaction interfaces, clustered into similar interfaces. These clusters have one representative, and their members have similar interfaces. In LISA, we created the a lookup table using cluster representatives. One challenging task is to find similarities between clusters and their members' structures. In production, a structure will be queried and LISA will try to find similarities between every cluster representative and query structure. Moreover, we need to query a chain and over the entire chain, we need to find smaller parts which are similar to various interfaces. Intuitively we need to find similarities such that a small part of an entire chain resembles a cluster representative. Hence, interfaces that we conducted tests using member interfaces' entire chains and tried to find their cluster representatives. This is because members of a cluster resembles the representative, thus if there is an actual similarity; LISA should find this resemblance over the entire chain.

In PIFACE, a large portion of the clusters have one member only, and they are represented by themselves. Finding similarities on these clusters are very easy. Since they are aligned exactly the same and the query structure has exactly the same construction with the cluster representative, LISA will instantly found because every single fragment will be matched and score will be significantly higher than other representatives.

4.1.1 First Experiment

To increase the robustness in our experiments, we selected clusters with different sizes which all have size larger than 10. After that, we run the algorithm. Finally, each cluster will have a similarity score. After these scores sorted from higher to lower, we applied top-n pruning. n is a hyper-parameter that can be defined on the preference. We have chosen different n which can be seen below. We sort the candidate correct results from the highest score to lower, we pruned the answers below n. If the correct answer is in the pruned list, we accepted that result as correct. Then we divided the number of correct results to number of tested structures and calculated accuracy. Results can be seen in Table 4.1.

Table 4.1: Results of first PIFACE experiment which cluster size is 10 or more for each test case.

Number of tested Chains: 200, Minimum Cluster Size:10				
Accuracy	Top 1 0.53	Top 10 0.77	Top 50 0.80	Top 100 0.81
	Top 500 0.84	Top 1000 0.85	Top 2500 0.87	Top 5000 0.89
	Top 10000 0.91	Top 15000 0.91	Top 25000 0.91	Top 35000 0.91
	Top 45000 0.91	Top 45208 0.91		
Avg Time	127 seconds			

4.1.2 Second Experiment

As second experiment, we randomly sampled chains from clusters with size bigger than 5. We applied the same pruning technique and accuracy metric. Results can

be seen in Table 4.2.

Table 4.2: Results of second PIFACE experiment which cluster size is 5 or more for each test case.

Number of tested Chains: 200, Minimum Cluster Size:5				
Accuracy	Top 1 0.64	Top 10 0.74	Top 50 0.78	Top 100 0.79
	Top 500 0.80	Top 1000 0.81	Top 2500 0.84	Top 5000 0.87
	Top 10000 0.89	Top 15000 0.90	Top 25000 0.90	Top 35000 0.90
	Top 45000 0.90	Top 45208 0.90		
Avg Time	126 seconds			

4.1.3 Third Experiment

As third experiment, we randomly selected chains from all clusters. Same pruning top-n approach and accuracy metric applied. Results can be seen in Table 4.3.

4.2 PRISM Experiments

As a second type of experiment, we conducted several benchmark tests on PRISM. In order to understand how well LISA can find a correct template for any arbitrary protein-protein interaction, we randomly sampled test subjects from PRISM results.

A single PRISM result represent an interaction between two target structures. Each interaction is calculated by using a template. After surface extraction of the targets, PRISM protocol uses Multiprot to find the possible templates for the given targets. Multiprot calculates any possible interaction between a target and list

Table 4.3: Results of third PIFACE experiment which cluster size is 1 or more for each test case.

Number of tested Chains: 200, Minimum Cluster Size:1				
Accuracy	Top 1 0.88	Top 10 0.95	Top 50 0.97	Top 100 0.97
	Top 500 0.97	Top 1000 0.98	Top 2500 0.98	Top 5000 0.98
	Top 10000 0.99	Top 15000 0.99	Top 25000 0.99	Top 35000 0.99
	Top 45000 0.99	Top 45208 0.99		
Avg Time	136 seconds			

of templates. LISA aims to put an upper bound to possible interactions with its filtering mechanism. To measure LISA's performance, we conducted a test upon existing PRISM predictions.

We applied weighted sampling among PRISM predictions and selected 160 different targets. Each of these target have several interactions among the results. Number of interactions vary from 1 to 2054. When we examined these interactions, we discovered that there are multiple interactions that are using the same interface. LISA does not need to find these templates more than once for a structure, so we took every single interface once as a correct answer. When we uniquely evaluate the results, number of possible correct answers varied from 1 to 231 for each structure.

Total number of interactions are 4155. We have considered an entry as a successful one is the binding energy score is below -5. In addition, we considered an interaction as successful if the template used in that interaction is in the top-1000 of LISA's result table. Different targets might have different number of interactions, hence different number of entries in PRISM results. For example, ("1p93", "AC")

structure has 1745 different entries in the database while ("4plm", "") has 1. Empty string in the tuple of structures name means entire structure has queried as target. Total number of interactions is 18676.

Another thing to mention is the redundancy of templates in the results. A template might be used more than once for different interactions. Hence, we only take unique elements and treat each unique template as one possible correct answer. The reason is, LISA will consider the same list of templates every time anyway. So the same set of 10.000 templates will be considered for the same structure every single time. This means we only need to find the correct answer once in LISA's top-10.000 list. When we remove the duplicates and eliminate redundancy, there were 4155 unique interfaces for different structures. LISA was able to find 2945 of them. Same interface might be in different structures' lists as the correct answer, the key point is that individual structures have their redundant lists before calculating the accuracy. LISA needs to find the same structure in different test subjects, because we are querying different structures, hence it needs to be found in both cases.

When we discarded the redundant elements from correct answers, number of possible templates varies from 1 to 231. When we got rid of the redundancy from the example ("1p93, "AC") above, we only have 40 different interfaces that LISA needs to find. This means the most widely queried target among our test subjects actually uses a very small set of interfaces comparing the number of total interactions it has on the PRISM database. On the other hand, ("1hx1", "B") has 633 interactions recorded in the PRISM, and it uses 231 different interfaces and having the most number of unique interfaces in PRISM among the test subjects.

Moreover, we also need to consider the strength of the interaction between targets. We specified different thresholds for the strength of interactions. If an interaction has a score below that threshold, we considered that interaction to be strong. Hence, we discarded interactions with a score higher than specified threshold.

Other parameter that needs to be specified is the LISA's top-n cutoff threshold, as we did in PIFACE experiments.

Results can be seen in Table 4.4.

Table 4.4: PRISM experiment results.

Number of tested Chains: 160, Weighted Sampling on PRISM Prediction Targets						
		Top-n				
		5000	10000	15000	20000	30000
Maximum Energy Threshold	-5	0.50	0.71	0.76	0.77	0.77
	-10	0.52	0.72	0.77	0.78	0.78
	-20	0.57	0.75	0.80	0.80	0.80
	-30	0.61	0.77	0.82	0.82	0.82
	-40	0.65	0.81	0.85	0.85	0.85
	-50	0.69	0.84	0.87	0.87	0.87

Chapter 5

RESULTS AND DISCUSSION

5.1 Wrong Answers on PIFACE Experiments

Even though results are promising, it is crucial to examine some of the wrong answers. Since LISA is designed to be a filtering algorithm, we can look at wrong results from a specific top-n case. For convenience, we will look at the wrong results from top-5000, since it is the ideal amount of filtering with satisfying accuracy score. This means we are going to examine the cases which do not have the correct result in the first 5000 scores after we sort the rankings.

5.1.1 First Experiment

First experiment yield to 89% accuracy. From 200 chains, there are 23 wrong answers. Let's take a closer look at those cases. List of wrong answers can be seen in Table 5.1. First column of the table represents the test cases. First element of the tuple is the structure and second element is the chain. Second column represents the correct answers with cluster size more than 10. Each correct answer has the following convention: first four letters represent the name of the structure of the correct representative, last two letters represents the chains of the interface. Some of the structures might have more than one answers, since one chain might be interacting with multiple chains. Also there might be some omitted answers, since we tried to find cluster with size more than 10. If a structure's chain is also in a cluster with less than 10 members, that cluster is omitted in this case, which increases the robustness of our experiment.

To examine cases, we picked the entire chain of the selected structure and generated the PDB of the correct answers' interface from PIFACE. Than, as a reference, we run TM-Align (Zhang and Skolnick, 2005) on the entire chain and generated

Table 5.1: Test cases which LISA could not find the correct cluster and their cluster representatives.

Wrong Case	Possible Answers
(”2WW9”, ”K”)	[”2GBJAB”]
(”1NLW”, ”B”)	[”3RWRGF”]
(”3Q2T”, ”B”)	[”3P6YIL”]
(”1YSF”, ”B”)	[”2W1TAB”]
(”3Q2T”, ”C”)	[”3P6YIL”]
(”1F5O”, ”C”)	[”1DEEGH”]
(”2AXG”, ”C”)	[”2F54AC”]
(”2WW9”, ”N”)	[”2GBJAB”]
(”2XPJ”, ”C”)	[”1W39AC”]
(”3ORB”, ”Y”)	[”2Y152X”]
(”2LMP”, ”N”)	[”2LMPKJ”]
(”2LMP”, ”O”)	[”2LMPKJ”]
(”2PWH”, ”A”)	[”2VPEAC”]
(”1UIR”, ”A”)	[”2O06AB”]
(”1YSF”, ”A”)	[”2W1TAB”]
(”3KND”, ”B”)	[”3FEYAC”]
(”1VS7”, ”U”)	[”2QP0KU”]
(”3RYF”, ”E”)	[”1Z2BED”, ”3HKCAE”]
(”1UIR”, ”B”)	[”2O06AB”]
(”2PWH”, ”B”)	[”2VPEAC”]
(”3JZT”, ”F”)	[”3ETIBD”]
(”2RKZ”, ”N”)	[”2RKZDP”]
(”3ORB”, ”T”)	[”2Y152X”]

interface structure from PIFACE same as in our experiments on LISA. Results can be seen below.

("2WW9", "K")

K chain of the structure 2WW9 could not be found by LISA. Chain K of the structure belongs to one side of the interface 2GBJAB. Hence, we will run TM-Align as a reference to find out if two structures are similar or not. TM-Align should find the similar chain and compute an alignment between the two structures. However, using the entire chain of 2WW9's K'th chain and 2GBJAB interface yield to a poor similarity result on TM-Align. Results for both side of the interface can be seen on the Table 5.2. First TM-score is normalized for the first structure, second TM-score is normalized for the second structure's lengths.

Table 5.2: TM-scores of ("2WW9", "K") aligned with 2GBJAB.

2GBJAB, A chain	TM-score= 0.14283, TM-score= 0.27544
2GBJAB, B chain	TM-score= 0.06274, TM-score= 0.39473

This results indicate that between ("2WW9", "K") and 2GBJAB. In order to have a significant similarity, TM-score should be higher than 0.5. TM-score can be between 0 and 1, more is better. Alignment of the two structures can be seen in the figure 5.1.

Figure 5.1: ("2WW9, "K") aligned with 2GBJAB's B chain.

Let's examine some of the LISA's findings. 25 strongest template candidates are listed in the Table A.1. Top match is 3P5TBO's B chain. We applied TM-Align for ("2WW9", "K") and the top match. Results can be seen in the Table 5.3.

Table 5.3: TM-scores of ("2WW9", "K") aligned with 3P5TBO's B chain.

3P5TBO, B chain	TM-score= 0.11831, TM-score= 0.21580

TM-scores show that the similarity between targets are at random. Their alignment can be seen in figure 5.2.

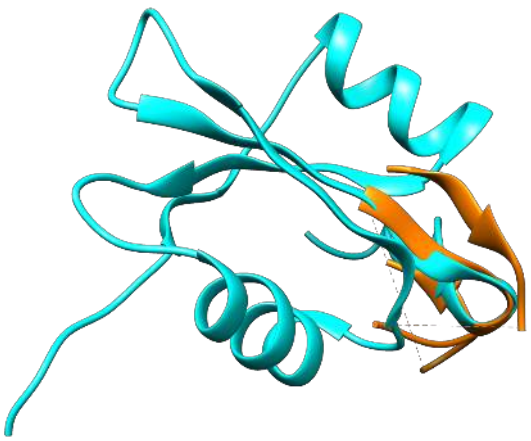


Figure 5.2: (”2WW9, ”K”) aligned with 3P5TBO’s B chain.

(”1NLW”, ”B”)

Correct answer for (”1NLW”, ”B”) is 3RWRGF. Results can be seen on the Table 5.4.

Table 5.4: TM-scores of (”1NLW”, ”B”) aligned with 3RWRGF.

3RWRGF, F chain	TM-score= 0.49068, TM-score= 0.58120
3RWRGF, G chain	TM-score= 0.49001, TM-score= 0.59456

For this case, we can say that between the test case and the correct result; there is a similarity. Alignment of the two structures can be seen in the figure 5.3.

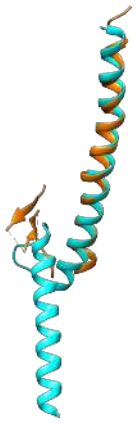


Figure 5.3: ("1NLW", "B") aligned with 3RWRGF's G chain.

Let's examine LISA's results. First 25 templates are given in Table A.2. We applied TM-Align to test subject and first result. TM-scores are given in the Table 5.5.

Table 5.5: TM-scores of ("1NLW", "B") aligned with 1E6CAB's B chain.

1E6CAB, B chain	TM-score= 0.17682, TM-score= 0.19261

TM-scores show that similarity between the structures are random. Their alignment can be seen in figure 5.4.

Figure 5.4: ("1NLW", "B") aligned with 1E6CAB's B chain.

("3Q2T", "B")

Correct answer for ("3Q2T", "B") is 3P6YIL. TM-scores can be seen in the table below.

Table 5.6: TM-scores of ("3Q2T", "B") aligned with 3P6YIL.

3P6YIL, I chain	TM-score= 0.09813, TM-score= 0.72038
3P6YIL, L chain	TM-score= 0.07016, TM-score= 0.18165

From the TM-scores, we can observe that ("3Q2T", "B") is similar to I chain of 3P6YIL. Alignment of the two structures can be seen in the figure 5.5.

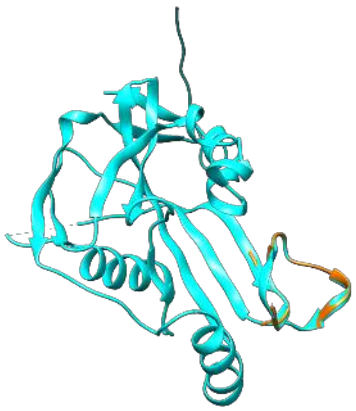


Figure 5.5: ("3Q2T", "B") aligned with 3P6YIL's I chain.

We examined this case further with LISA's results. Scores from highest 25 template candidates are given in the Table A.3. It appears that LISA found several interfaces with high scores. Let's examine the first structure.

Table 5.7: TM-scores of ("3Q2T", "B") aligned with 3Q2SAB's B chain.

3Q2SAB, B chain	TM-score= 0.29503, TM-score= 0.99253

LISA found a more similar structure than PIFACE match according to TM-Align. We can examine it in the figure 5.6.

Figure 5.6: (”3Q2T”, ”B”) aligned with 3Q2SAB’s B chain.

It is clear that template candidate that LISA found is more similar and a bigger interface. It is also possible to use the template from PIFACE, since their regions are different than each other. One thing to consider is LISA is not successful with small interfaces, which will be discussed in detail in the next sections.

(”1YSF”, ”B”)

Correct answer for (”1YSF”, ”B”) is 2W1TAB. TM-scores can be seen in the Table 5.8.

Table 5.8: TM-scores of (”1YSF”, ”B”) aligned with 2W1TAB.

2W1TAB, A chain	TM-score= 0.78111, TM-score= 0.75794
2W1TAB, B chain	TM-score= 0.74471, TM-score= 0.80301

Apparently, (”1YSF”, ”B”) and 2W1TAB have similar structures. Particularly, B chain of 2W1TAB has the highest score of all TM-scores. Alignment of the (”1YSF”, ”B”) and 2W1TAB’s B chain can be seen in the figure 5.7.

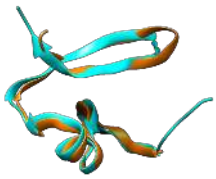


Figure 5.7: ("1YSF", "B") aligned with 2W1TAB's B chain.

We examined LISA's results below. First 25 candidates are listed in the Table A.4. Best candidate is 3a17AD D.int according to LISA, but score is not very high. Let us examine it with TM-Align in the table below.

Table 5.9: TM-scores of ("1YSF", "B") aligned with 3A17AD's D chain.

3A17AD, D chain	TM-score= 0.23835, TM-score= 0.28805

It is clear that LISA's candidate structure is not similar to our test case. We can see the two structures in the figure 5.8.

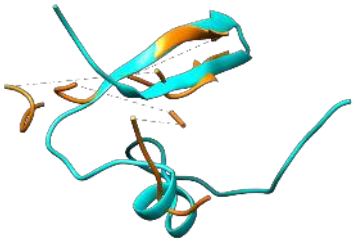


Figure 5.8: ("1YSF", "B") aligned with 3A17AD's D chain.

For this case, LISA could not find a successful match.

("3Q2T", "C")

("3Q2T", "C") is under the cluster where the 3P6YIL is the representative. Results of the TM-Align can be seen in the Table 5.10.

Table 5.10: TM-scores of ("3Q2T", "C") aligned with 3P6YIL.

3P6YIL, I chain	TM-score= 0.15029, TM-score= 0.23796
3P6YIL, L chain	TM-score= 0.25460, TM-score= 0.81914

It is clear that ("3Q2T", "C") has a similarity with 3P6YIL's, L chain. Their alignment can be seen on the figure 5.9.

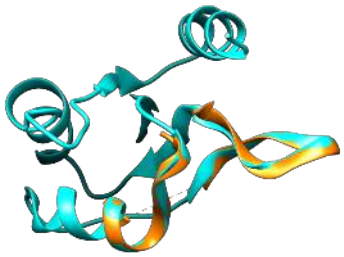


Figure 5.9: ("3Q2T", "C") aligned with 3P6YIL's L chain.

Let's examine LISA's results. First 25 case is given in the Table A.5.

LISA did not find a strong candidate for this test case. Let's examine the best case in TM-Align.

Table 5.11: TM-scores of ("3Q2T", "C") aligned with 1NPPAB's A chain.

1NPPAB, A chain	TM-score= 0.21388, TM-score= 0.25467

TM-Align does not indicate a similarity for this case either. We can investigate the alignment in figure 5.10.

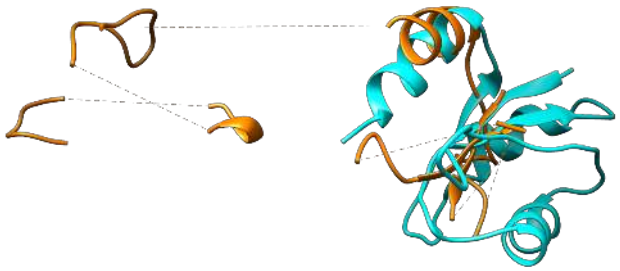


Figure 5.10: ("3Q2T", "C") aligned with 1NPPAB's A chain.

Correct answers that comes from PIFACE is a small structure. This issue will be discussed in the further sections.

("1F5O", "C")

("1F5O", "C") has a similar interface with 1DEEGH according to PIFACE. Their alignment scores can be seen in the Table 5.12.

Table 5.12: TM-scores of ("1F5O", "C") aligned with 1DEEGH.

1DEEGH, G chain	TM-score= 0.16415, TM-score= 0.52608
1DEEGH, H chain	TM-score= 0.14182, TM-score= 0.47401

G chain of 1DEEGH seems to be about within the same fold. Superimposition of the two structures can be seen in the figure 5.11.

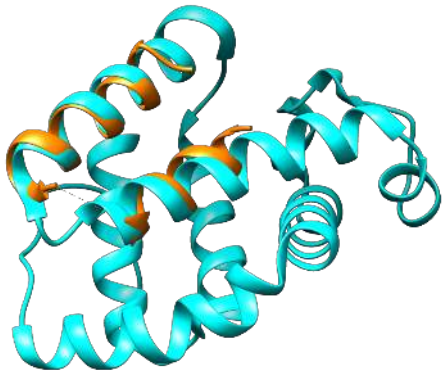


Figure 5.11: ("1F5O", "C") aligned with 1DEEGH's G chain.

Though their structures look quite similar, we examined further and derived results from LISA. Results can be seen in the table A.6. It is clear that LISA calculated a very high score for "3lhbCH C.int" interface. This means the interface 3LHBCH is similar with ("1F5O", "C") from the C chain's side. It is necessary to examine the TM-scores for this structure. Scores can be seen from the table below.

Table 5.13: TM-scores of ("1F5O", "C") aligned with 3LHBCH's C chain.

3LHBCH, C chain	TM-score= 0.24140, TM-score= 0.99336

TM-Align algorithm determines a better similarity between the structure that is determined by LISA as the most similar structure. 0.9936 is a significantly higher score. This results show that LISA was able to find an almost identical case, whereas the correct answer is barely in the same fold. Their alignments can be seen on the figure 5.12.

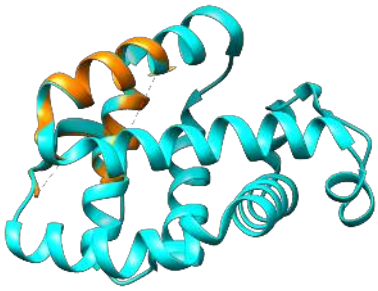


Figure 5.12: ("1F5O", "C") aligned with 3LHBCH's C chain.

Though correct result coming from PIFACE and LISA's derived results are similar, they are not in the same cluster, moreover, 1F5O is not included in the cluster that LISA derived in the PIFACE. Even though similar parts are not the same, one could argue that using an interface with higher TM-Score might yield to more accurate results in further template based protein protein interaction processes.

("2AXG", "C")

Correct answer for ("2AXG", "C") is 2F54AC. TM-scores can be seen in the Table 5.14.

Table 5.14: TM-scores of ("2AXG", "C") aligned with 2F54AC.

2F54AC, A chain	TM-score= 0.29667, TM-score= 0.03571
2F54AC, C chain	TM-score= 0.37264, TM-score= 0.41404

For this case, TM-scores does not look very promising. All scores are below 0.5, which tells us that there is no similarity between the two structures. Their alignment can be seen in the figure 5.13.

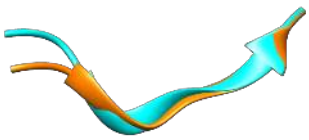


Figure 5.13: ("2AXG", "C") aligned with 2F54AC's C chain.

For this case, it seems that neither TM-Align nor LISA could not find the similarity. This might due to the fact that structures are small. We examined the outputs from LISA, turns out there are not any match from any interface templates. This might be due to the fact that the size is extremely small.

("2WW9", "N")

("2WW9", "N") has the correct answer 2GBJAB. We will examine the TM-score results below.

Table 5.15: TM-scores of ("2WW9", "N") aligned with 2GBJAB.

2GBJAB, A chain	TM-score= 0.13643, TM-score= 0.12332
2GBJAB, B chain	TM-score= 0.10704, TM-score= 0.10256

TM-scores are not satisfactory for the structures. Their alignment can be seen in the figure 5.14, and it is clear that they are not similar to each other.

Figure 5.14: ("2WW9", "N") aligned with 2GBJAB's A chain.

Let's examine LISA's findings. Scores of the first 25 structures found by LISA are given in the Table A.7. Score of the first structure indicates a significant similarity between query structure and itself. Let's examine the two in TM-Align. Scores are given below.

Table 5.16: TM-scores of ("2WW9", "N") aligned with 2WW9AN's N chain.

2WW9AN, N chain	TM-score= 0.17391, TM-score= 1.00000

As they are the same structure TM-score is exactly 1. In this case, naturally LISA was able to find a correct result. The reason 2WW9AN was not considered as an answer in out evaluations is the number of members in the 2WW9AN is less than 10. There are 2 members in 2WW9AN, whereas there are 15 members in 2WW9KN cluster. Alignment of the structures can be seen in the figure 5.15.

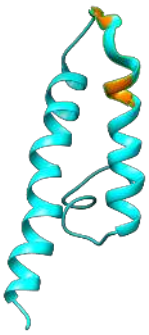


Figure 5.15: ("2WW9", "N") aligned with 2WW9AN's N chain.

("2XPJ", "C")

("2XPJ", "C") chain is in the cluster where 1W39AC is the representative.

Table 5.17: TM-scores of ("2XPJ", "C") aligned with 1W39AC.

1W39AC, A chain	TM-score= 0.09795, TM-score= 0.35960
1W39AC, C chain	TM-score= 0.22701, TM-score= 0.88162

TM-Align shows that structures are very similar to each other. Alignment can be seen in the figure 5.16.

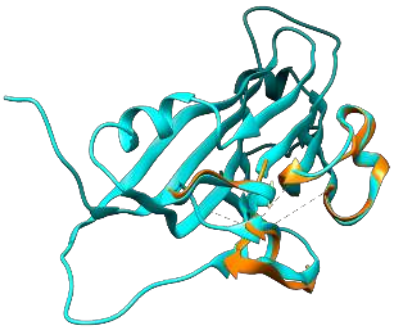


Figure 5.16: ("2XPJ", "C") aligned with 1W39AC's C chain.

Let's examine the results from LISA. Top 25 candidates are given in the Table A.8. Let's examine the first structure from the results in the Table 5.18.

Table 5.18: TM-scores of ("2XPJ", "C") aligned with 1YQ2AF's A chain.

1YQ2AF, A chain	TM-score= 0.09541, TM-score= 0.23540

TM-Align does not indicate a strong similarity between the two structures. Their TM-score is below 0.3, which declares that their similarity is random. Their alignment can be seen in the figure 5.17 below.

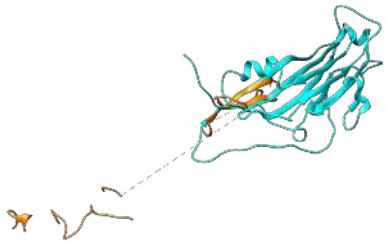


Figure 5.17: ("2XPJ", "C") aligned with 1YQ2AF's A chain.

("3ORB", "Y")

Obsolete PDB file.

("2LMP", "N")

("2LMP", "N") matches with 2LMPKJ. TM-scores are in the Table 5.19.

Table 5.19: TM-scores of ("2LMP", "N") aligned with 2LMPKJ.

2LMPKJ, K chain	TM-score= 0.41408, TM-score= 0.41408
2LMPKJ, J chain	TM-score= 0.46333, TM-score= 0.46333

Score is below 0.5 which means similarity is not that rigorous for this structure according to TM-Align. Their alignment can be seen in the figure 5.18 below.

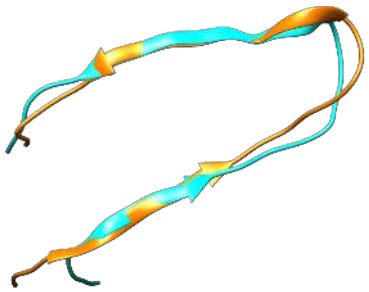


Figure 5.18: ("2LMP", "N") aligned with 2LMPKJ's J chain.

Though their structure has a similarity, neither TM-Align nor LISA could find it. Let's examine the first 25 candidate answers from LISA in the Table A.9. From the results, we can derive the conclusion that LISA could not find a similarity among any possible interfaces. Let's use TM-Align to align the structure with the highest score and out test subject.

Table 5.20: TM-scores of ("2LMP", "N") aligned with 1TL8AD's A chain.

1TL8AD, A chain	TM-score= 0.15980, TM-score= 0.21684

TM-Align score shows a random structural similarity between the targets. Let's examine the alignment in the figure 5.19.

Figure 5.19: ("2LMP", "N") aligned with 1TL8AD's A chain.

From the image it is clear that similarity is quite low, but we can achieve the same conclusion by examining the scores as well.

("2LMP", "O")

("2LMP", "O") has the correct answer 2LMPKJ. This means both ("2LMP", "O") and ("2LMP", "N") have an interface between them, where their cluster representative is 2LMPKJ. Their alignment can be seen in the Table 5.21.

Table 5.21: TM-scores of ("2LMP", "O") aligned with 2LMPKJ.

2LMPKJ, K chain	TM-score= 0.42909, TM-score= 0.42909
2LMPKJ, J chain	TM-score= 0.50699, TM-score= 0.50699

For this side of the interface, TM-Align determines that there is a similarity. Their alignments can be seen in the figure 5.20 below.

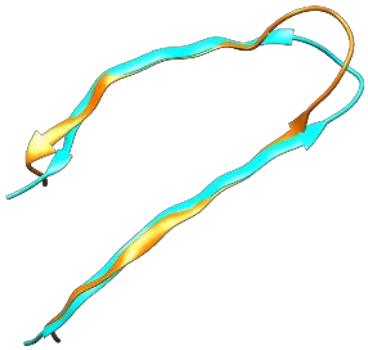


Figure 5.20: ("2LMP", "O") aligned with 2LMPKJ's J chain.

Both side of the interfaces have very similar structure. Though this is a structure with β strands, hence it is harder to align, especially around the loop regions. Examination of LISA's results are given in the Table A.10. Results indicate that there is not a strong similarity between any of the possible templates and the test structure. Yet, let's examine the strongest match.

Table 5.22: TM-scores of ("2LMP", "O") aligned with 1QH2AB, A chain.

1QH2AB, A chain	TM-score= 0.22110, TM-score= 0.22168

TM-Align score shows a random structural similarity between the targets. Let's examine the alignment in the figure 5.21.

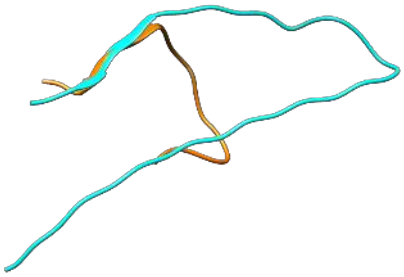


Figure 5.21: ("2LMP", "O") aligned with 1QH2AB's A chain.

("2PWH", "A")

("2PWH", "A") is in the cluster of 2VPEAC. Their TM-scores can be seen below.

Table 5.23: TM-scores of ("2PWH", "A") aligned with 2VPEAC.

2VPEAC, A chain	TM-score= 0.02065, TM-score= 0.37499
2VPEAC, C chain	TM-score= 0.02066, TM-score= 0.37984

It is clear that they are not similar structures. Their alignment can be seen in the figure 5.22.

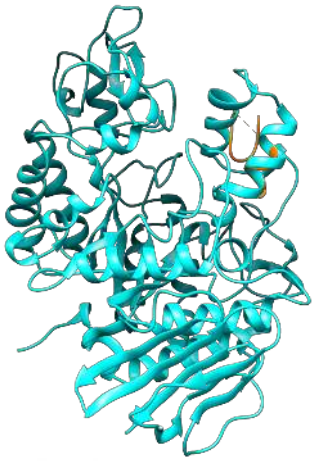


Figure 5.22: ("2PWH", "A") aligned with 2VPEAC's C chain.

It is clear that C chain of the 2VPEAC is a very small structure. TM-Align could not manage to succeed aligning two structures. Let's examine LISA's findings about the test structure in the Table A.11. Specifically first structure show a similarity. TM-scores between the test structure and highest ranked template are given in the table below.

Table 5.24: TM-scores of ("2PWH", "A") aligned with 1DYOAB, A chain.

1DYOAB, A chain	TM-score= 0.02676, TM-score= 0.22886

TM-scores shows there is a random structural similarity between the targets. Alignment can be seen in the figure 5.23.

Figure 5.23: ("2PWH", "A") aligned with 1DYOAB's A chain.

Even though the TM-score is under 0.3, from the figure 5.23, we can see that loop region of the test structure aligned with the interface template.

("1UIR", "A")

("1UIR", "A") and 2O06AB are similar according to PIFACE. Their TM-scores and alignment are given in the Table 5.25.

Table 5.25: TM-scores of ("1UIR", "A") aligned with 2O06AB.

2O06AB, A chain	TM-score= 0.23439, TM-score= 0.73550
2O06AB, B chain	TM-score= 0.23373, TM-score= 0.72223

Structures are in about the same fold. Their alignments are given below 5.24.

Figure 5.24: ("1UIR", "A") aligned with 2O06AB's A chain.

For this case we can observe that both structures are similar. Even though correct answer is not in LISA's results, we can observe its strongest candidates in the Table A.12. Let's examine the strongest possible candidate 1QWTAB's A chain below.

Table 5.26: TM-scores of ("1UIR", "A") aligned with 1QWTAB, A chain.

1QWTAB, A chain	TM-score= 0.06675, TM-score= 0.35120

TM-scores does not show a similarity. Let's examine the alignment in the figure 5.25.

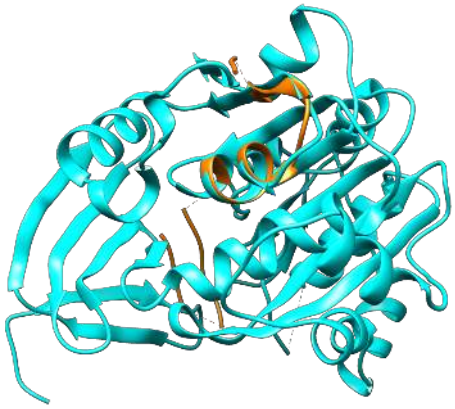


Figure 5.25: ("1UIR", "A") aligned with 1QWTAB's A chain.

("1YSF", "A")

("1YSF", "A") and 2W1TAB are matched as the test case and the correct answer. Case study of the two structures are given in the Table 5.27.

Table 5.27: TM-scores of ("1YSF", "A") aligned with 2W1TAB.

2W1TAB, A chain	TM-score= 0.78103, TM-score= 0.75787
2W1TAB, B chain	TM-score= 0.74445, TM-score= 0.80304

TM-Align determines a strong similarity for this case. Alignment is given on the figure 5.26.

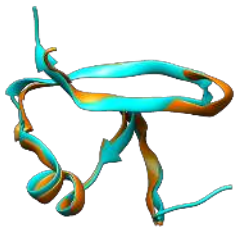


Figure 5.26: ("1YSF", "A") aligned with 2W1TAB's B chain.

25 of the LISA's favourite candidates are given in the Table A.13. TM-scores between ("1YSF", "A") and 1LL4AB's B chain are given below.

Table 5.28: TM-scores of ("1YSF", "A") aligned with 1LL4AB, B chain.

1LL4AB, B chain	TM-score= 0.14858, TM-score= 0.14976

TM-scores indicate that the similarity is at random. Their alignment can be seen in figure 5.27.

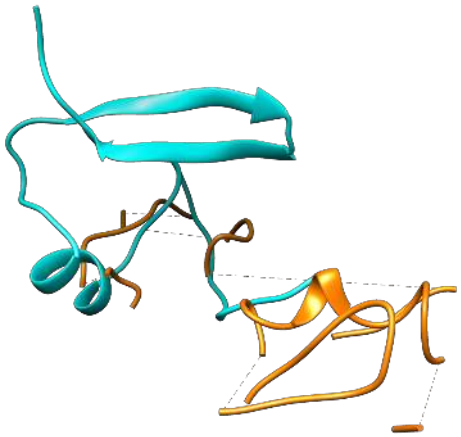


Figure 5.27: (”1YSF”, ”A”) aligned with 1LL4AB’s B chain.

It is clear that structures did not align very well.

(”3KND”, ”B”)

We examined (”3KND”, ”B”) and 3FEYAC below.

Table 5.29: TM-scores of (”3KND”, ”B”) aligned with 3FEYAC.

3FEYAC, A chain	TM-score= 0.45935, TM-score= 0.38167
3FEYAC, C chain	TM-score= 0.22876, TM-score= 0.04328

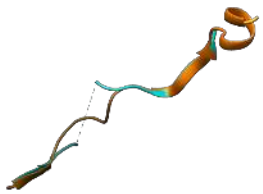


Figure 5.28: ("3KND", "B") aligned with 3FEYAC's A chain.

It is clear that both structures are similar to each other. However, LISA could not derive any possible candidates for the test subject.

("1VS7", "U")

Obsolete PDB file.

("3RYF", "E")

("3RYF", "E") has two correct answers: 1Z2BED and 3HKCAE. We examined both correct answers individually below in Table 5.30 and Table 5.31.

1Z2BED Alignment of 1Z2BED is given below.

Table 5.30: TM-scores of ("3RYF", "E") aligned with 1Z2BED.

1Z2BED, E chain	TM-score= 0.11693, TM-score= 0.24698
1Z2BED, D chain	TM-score= 0.18102, TM-score= 0.74958

0.74 TM-score indicates a similarity between the two structures. Their alignments can be seen on the figure 5.29.

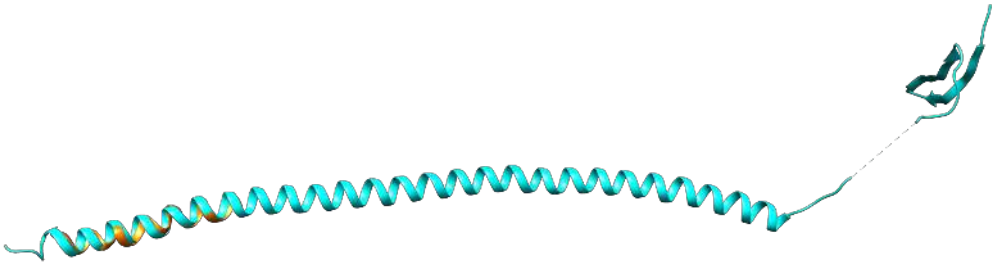


Figure 5.29: ("3RYF", "E") aligned with 1Z2BED's D chain.

Even though ("3RYF", "E") is a large structure, we can observe a similarity in the image as well.

3HKCAE Alignment of 3HKCAE is given below.

Table 5.31: TM-scores of ("3RYF", "E") aligned with 3HKCAE.

3HKCAE, A chain	TM-score= 0.11929, TM-score= 0.17121
3HKCAE, E chain	TM-score= 0.34232, TM-score= 0.83027

0.83 TM-score indicates a similarity between the two structures. Their alignments can be seen on the figure 5.30.

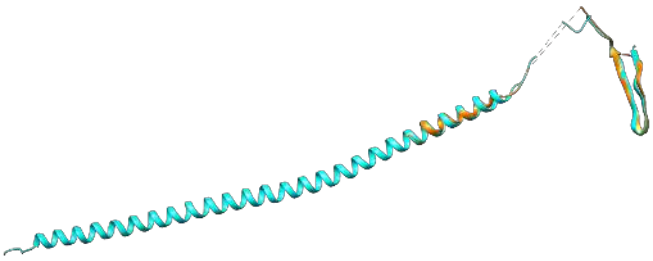


Figure 5.30: ("3RYF", "E") aligned with 3HKCAE's E chain.

Two structures are apparently similar to each other, which can be seen in the image.

Let's examine LISA's results in the Table A.14. TM-scores between the first structure and the test subject are given below.

Table 5.32: TM-scores of ("3RYF", "E") aligned with 1HQRCD, D chain.

1HQRCD, D chain	TM-score= 0.04377, TM-score= 0.19784

Alignment of ("3RYF", "E") and 1HQRCD's D chain is given in figure 5.31.

Figure 5.31: (”3RYF”, ”E”) aligned with 1HQRCD’s D chain.

(”1UIR”, ”B”)

Correct answer for (”1UIR”, ”B”) is 2O06AB. We examined it in Table 5.33.

Table 5.33: TM-scores of (”1UIR”, ”B”) aligned with 2O06AB.

2O06AB, A chain	TM-score= 0.23307, TM-score= 0.73785
2O06AB, B chain	TM-score= 0.23231, TM-score= 0.72412

TM-Align indicates a similarity between the structures. Their alignment can be seen in the figure 5.32.

Figure 5.32: ("1UIR", "B") aligned with 2O06AB's A chain.

Similarity can be seen between the two structures in the figure. LISA's results can on the other hand, can be seen in Table A.15 for first 25 strongest template candidates. First structure and test subject are given to TM-Align. Scores can be seen below.

Table 5.34: TM-scores of ("1UIR", "B") aligned with 2BYUAI, A chain.

2BYUAI, A chain	TM-score= 0.06238, TM-score= 0.32196

TM-scores indicates a that similarity is just above random. Alignment of targets can be seen in the figure 5.33.

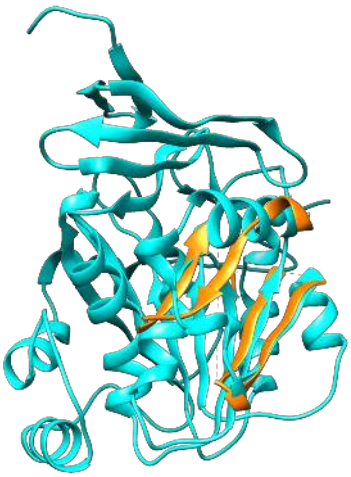


Figure 5.33: ("1UIR", "B") aligned with 2BYUAI's A chain.

("2PWH", "B")

("2PWH", "B") is in the cluster 2VPEAC. We examined it in Table 5.35.

Table 5.35: TM-scores of ("2PWH", "B") aligned with 2VPEAC.

2VPEAC, A chain	TM-score= 0.02067, TM-score= 0.37048
2VPEAC, C chain	TM-score= 0.02068, TM-score= 0.37157

TM-Align could not find a tangible alignment between the two structures. Alignment can be seen in the figure 5.34.

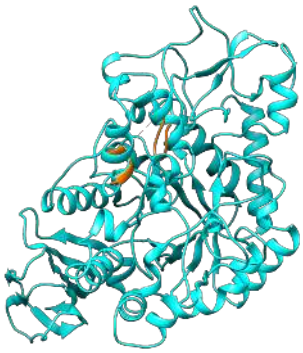


Figure 5.34: ("2PWH", "B") aligned with 2VPEAC's C chain.

There is no visible alignment between the chain and the interface. LISA's top 25 results are given in the Table A.16. TM-scores between the first structure and test subject are given in the Table 5.36.

Table 5.36: TM-scores of ("2PWH", "B") aligned with 2WZJCD, C chain.

2WZJCD, C chain	TM-score= 0.06830, TM-score= 0.36171

TM-score is just above random structural similarity. Alignment of the targets can be seen in figure 5.35.

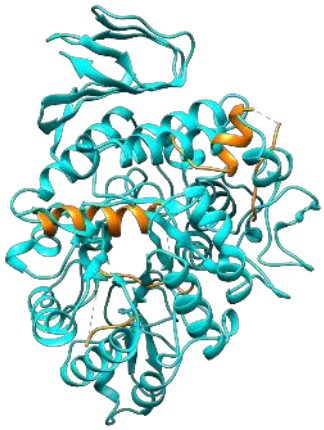


Figure 5.35: (”2PWH”, ”B”) aligned with 2WZJCD’s C chain.

(”3JZT”, ”F”)

We examined (”3JZT”, ”F”) and 3ETIBD below.

Table 5.37: TM-scores of (”3JZT”, ”F”) aligned with 3ETIBD.

3ETIBD, B chain	TM-score= 0.14072, TM-score= 0.66956
3ETIBD, D chain	TM-score= 0.08143, TM-score= 0.58740

TM-Align indicates a similarity. Their alignment can be seen in the figure 5.36.

Figure 5.36: ("3JZT", "F") aligned with 3ETIBD's B chain.

Now let's examine the 25 strongest matching candidates from LISA. Results can be seen in Table A.17. TM-scores of strongest candidate are given in Table 5.38.

Table 5.38: TM-scores of ("3JZT", "F") aligned with 1RF5BC, B chain.

1RF5BC, B chain	TM-score= 0.11485, TM-score= 0.25358

TM-score did not yield to a promising result. Below 0.3 TM-score implies a random structural similarity between the targets. Alignment of the targets can be seen in figure 5.37.

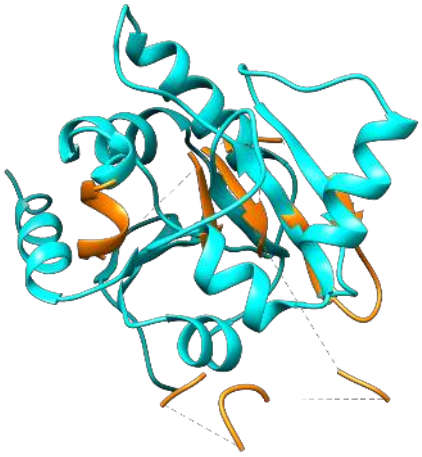


Figure 5.37: ("3JZT", "F") aligned with 1RF5BC's B chain.

("2RKZ", "N")

("2RKZ", "N") and 2RKZDP examined below.

Table 5.39: TM-scores of ("2RKZ", "N") aligned with 2RKZDP.

2RKZDP, D chain	TM-score= 0.21445, TM-score= 0.12535
2RKZDP, P chain	TM-score= 0.48873, TM-score= 0.48873

Having a TM-score below 0.5 show that there is no similarity between the two structures. However, they are close to 0.5 for P chain of 2RKZDP. Alignment of the two structures can be seen on the figure 5.38.

Figure 5.38: ("2RKZ", "N") aligned with 2RKZDP's P chain.

LISA did not find any similarity among any of the interfaces.

("3ORB", "T")

Obsolete PDB file.

5.2 Discussion Over Results

LISA is a promising algorithm for filtering the number of possible alignments between a given list of templates and a query structure. Statistically, it performs well and generates a shortened template list for structural alignment. However, when we investigate the wrong results from PIFACE tests, we found out that it performs specifically poorly on small interfaces, interfaces that are exist only on loop regions and single Alpha-helices. Beta-sheets are LISA's another weakness, since they are wide and do not contain enough number of residues to calculate enough fragments. LISA can filter dense interfaces with enough number of residues easily including eligible interfaces for PPI prediction. Other test cases could be found in the top-5000 for PIFACE tests.

PRISM tests are much harder, considering correct answers are not clustered by any authority, and there can be arbitrary template choices in that manner. Yet LISA

was able to perform 0.71 accuracy by finding most of the answers in its top-10000 table.

Chapter 6

CONCLUSION

Here we present LISA, a fast filtering algorithm for reducing the number of possible alignment candidates over a given data set. It generates sets of pieces called fragments from given interface structures and stores them into a hash table. After extracting fragments from an arbitrary query structure, by using feature extraction and quantization methods LISA can find similarities between the query structure and interfaces that stored in the hash table. Finally, applying transformations on each matched fragment pairs yield to another hash table, leading score calculation for each interface. After sorting the scores, LISA is limiting the number of matches by applying a cutoff threshold in which determined empirically, generates a filtered list that contains strong candidates for structural alignment and docking methods.

BIBLIOGRAPHY

Bahar, I., Jernigan, R. L., and Dill, K. A. (2017). *Protein actions: Principles and modeling*. Garland Science, Taylor &; Francis Group, LLC.

Balcı, A. T. (2018). *Fast Screening Algorithms for Protein Interface Structural Alignments*. PhD thesis, Koç University.

Baspinar, A., Cukuroglu, E., Nussinov, R., Keskin, O., and Gursoy, A. (2014). Prism: a web server and repository for prediction of protein–protein interactions and modeling their 3d complexes. *Nucleic acids research*, 42(W1):W285–W289.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The protein data bank. *Nucleic acids research*, 28(1):235–242.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98.

Braun, P. and Gingras, A.-C. (2012). History of protein–protein interactions: From egg-white to complex networks. *Proteomics*, 12(10):1478–1498.

Burley, S. K., Berman, H. M., Kleywegt, G. J., Markley, J. L., Nakamura, H., and Velankar, S. (2017). Protein data bank (pdb): the single global macromolecular structure archive. *Protein Crystallography*, pages 627–641.

Cukuroglu, E., Gursoy, A., Nussinov, R., and Keskin, O. (2014). Non-redundant unique interface structures as templates for modeling protein interactions. *PloS one*, 9(1):e86738.

Deng, L., Zhong, G., Liu, C., Luo, J., and Liu, H. (2019). Madoka: an ultra-fast approach for large-scale protein structure similarity searching. *BMC bioinformatics*, 20(19):1–10.

Deng, M., Zhang, K., Mehta, S., Chen, T., and Sun, F. (2002). Prediction of protein function using protein-protein interaction data. In *Proceedings. IEEE Computer Society Bioinformatics Conference*, pages 197–206.

Estrin, M. and Wolfson, H. J. (2017). SnapDock—template-based docking by Geometric Hashing. *Bioinformatics*, 33(14):i30–i36.

Fischer, D., Nussinov, R., and Wolfson, H. J. (1992). 3-d substructure matching in protein molecules. In *Annual Symposium on Combinatorial Pattern Matching*, pages 136–150. Springer.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.

Greener, J. G., Selvaraj, J., and Ward, B. J. (2020). BioStructures.jl: read, write and manipulate macromolecular structures in Julia. *Bioinformatics*, 36(14):4206–4207.

Grosdidier, A., Zoete, V., and Michielin, O. (2011). Swissdock, a protein-small molecule docking web service based on eadock dss. *Nucleic acids research*, 39(suppl_2):W270–W277.

Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828.

Lu, Z., Zhao, Z., and Fu, B. (2010). Efficient protein alignment algorithm for protein search. *BMC bioinformatics*, 11(1):1–10.

Mashiach, E., Nussinov, R., and Wolfson, H. J. (2010). Fiberdock: flexible induced-fit backbone refinement in molecular docking. *Proteins: Structure, Function, and Bioinformatics*, 78(6):1503–1519.

Nussinov, R. and Wolfson, H. J. (1991). Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proceedings of the National Academy of Sciences*, 88(23):10495–10499.

Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., and Ferrin, T. E. (2004). Ucsf chimera—a visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612.

Porter, K. A., Desta, I., Kozakov, D., and Vajda, S. (2019). What method to use for protein–protein docking? *Current opinion in structural biology*, 55:1–7.

Rao, V. S., Srinivas, K., Sujini, G., and Kumar, G. (2014). Protein-protein interaction detection: methods and analysis. *International journal of proteomics*, 2014.

Shatsky, M., Nussinov, R., and Wolfson, H. J. (2002). Multiprot—a multiple protein structural alignment algorithm. In *International Workshop on Algorithms in Bioinformatics*, pages 235–250. Springer.

Stockman, G. (1987). Object recognition and localization via pose clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387.

Sun, H., Sacan, A., Ferhatosmanoglu, H., and Wang, Y. (2011). Smolign: a spatial motifs-based protein multiple structural alignment method. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(1):249–261.

Szilagyi, A. and Zhang, Y. (2014). Template-based structure modeling of protein–protein interactions. *Current Opinion in Structural Biology*, 24:10–23. Folding and binding / Nucleic acids and their protein complexes.

Tuncbag, N., Gursoy, A., and Keskin, O. (2009). Identification of computational hot spots in protein interfaces: combining solvent accessibility and inter-residue potentials improves the accuracy. *Bioinformatics*, 25(12):1513–1520.

Tuncbag, N., Gursoy, A., Nussinov, R., and Keskin, O. (2011). Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using prism. *Nature protocols*, 6(9):1341–1354.

Wolfson, H. J. and Rigoutsos, I. (1997). Geometric hashing: an overview. *IEEE Computational Science and Engineering*, 4(4):10–21.

Zhang, Y. and Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710.

Zhang, Y. and Skolnick, J. (2005). Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309.

Appendix A

Table A.1: LISA's top 25 results in descending order for ("2WW9", "K") with similarity scores.

9.02469 => "3p5tBO_B.int"
9.01235 => "3ndyEF_F.int"
8.11323 => "3eh2AC_C.int"
6.77372 => "1xq7AB_A.int"
6.29167 => "2ou1BC_C.int"
6.28279 => "3qitBD_D.int"
6.275 => "2iagAB_A.int"
6.27299 => "1ybdAB_B.int"
6.27247 => "2bz9AB_B.int"
6.25862 => "2x8kAB_A.int"
6.14181 => "2ivoAB_B.int"
6.06965 => "2bz0AB_A.int"
6.03125 => "3iwuBF_F.int"
6.0229 => "3doiAB_B.int"
6.02143 => "3gzbAC_C.int"
6.01734 => "1z0aCD_D.int"
6.01288 => "2b2nAB_B.int"
4.58933 => "1dgrBV_B.int"
4.54082 => "1ahwBC_B.int"
4.52649 => "3ituCD_D.int"
4.51036 => "3iynDR_D.int"
4.50928 => "3cq9CD_C.int"
4.5 => "2of5KL_K.int"
4.28667 => "3ijsBD_B.int"
4.24359 => "1n0xPR_P.int"

Table A.2: LISA's top 25 results in descending order for ("1NLW", "B") with similarity scores.

10.315 => "1e6cAB_B.int"
8.6684 => "1q2vBD_D.int"
8.07692 => "1a02FN_F.int"
8.03226 => "2grvAC_A.int"
7.79884 => "1mbxAD_D.int"
7.5 => "3broAD_D.int"
7.25405 => "2z4rBC_B.int"
6.66864 => "3s5uBC_C.int"
6.37879 => "1w72DI_D.int"
6.29826 => "3a29BD_D.int"
6.2617 => "3dedAB_B.int"
6.2601 => "2js3AB_B.int"
6.08333 => "3h52BD_D.int"
5.88214 => "3favAC_A.int"
5.63132 => "1nh2BD_D.int"
5.56667 => "3bkdGH_G.int"
5.1029 => "1n2mEF_E.int"
4.53704 => "1wwzAB_B.int"
4.52667 => "2ofjAB_B.int"
4.51887 => "2guzIJ_I.int"
4.40948 => "2nxoBC_B.int"
4.33333 => "3hfaUV_V.int"
4.31429 => "1niwAB_B.int"
4.28068 => "1tmxAB_A.int"
4.22433 => "2gyqAB_A.int"

Table A.3: LISA's top 25 results in descending order for ("3Q2T", "B") with similarity scores.

54374.0 =>"3q2sAB_B.int"
633.774 =>"3p5tAB_B.int"
116.789 =>"3q2sAB_A.int"
51.0177 =>"3p5tAB_A.int"
37.8159 =>"3fmbAB_B.int"
18.7796 =>"3l7qGI_G.int"
15.4038 =>"1d3yAB_A.int"
14.0285 =>"3m8fAB_B.int"
12.2734 =>"1gl4AB_A.int"
12.1472 =>"1oidAB_A.int"
11.6052 =>"2isnAB_B.int"
10.6913 =>"2dcyBC_B.int"
9.17149 =>"2y7qBD_B.int"
8.57398 =>"1kkeAC_C.int"
8.12101 =>"3rmhAB_B.int"
8.02649 =>"2bmcBD_D.int"
8.01493 =>"2ov2FN_F.int"
6.56871 =>"2isnAB_A.int"
6.42874 =>"2o2aBC_C.int"
6.39286 =>"3o3vBC_C.int"
6.28158 =>"1wyzBD_B.int"
6.27439 =>"2x54NV_N.int"
6.27273 =>"3i3cBC_C.int"
6.26667 =>"3hvlAB_B.int"
6.26163 =>"2r0hAB_B.int"

Table A.4: LISA's top 25 results in descending order for ("1YSF", "B") with similarity scores.

9.01481 => "3a17AD_D.int"
6.42994 => "3lqmAB_B.int"
6.41089 => "3izwJV_J.int"
6.25485 => "3g7gCG_C.int"
6.25412 => "3emfAB_A.int"
4.58513 => "2i99AB_B.int"
4.1901 => "1s4iBD_D.int"
4.18707 => "1j6rAB_B.int"
4.18553 => "3i12BC_C.int"
4.17604 => "3demAB_A.int"
4.17519 => "3demAB_B.int"
4.17233 => "3q34BD_D.int"
4.03279 => "3qitBD_D.int"
4.0241 => "1h0jAB_A.int"
4.01481 => "1wwhAB_A.int"
4.00833 => "1ll4AB_B.int"
4.008 => "2ayoAB_B.int"
4.00627 => "3ofgAB_B.int"
4.00475 => "1iq6AB_B.int"
3.61055 => "3bcwAB_B.int"
3.175 => "1qa7AB_B.int"
3.1553 => "2gfgAC_A.int"
3.15132 => "1nkqBF_B.int"
3.1491 => "1ptoIJ_I.int"
3.14335 => "3ob0HM_M.int"

Table A.5: LISA's top 25 results in descending order for ("3Q2T", "C") with similarity scores.

12.121 => "1nppAB_A.int"
9.02469 => "3rv5CD_C.int"
8.17694 => "2ffsAB_B.int"
8.03509 => "2z2cAD_A.int"
6.29082 => "2a7kCI_I.int"
6.27597 => "2a7kFI_F.int"
6.2741 => "2g9tQR_Q.int"
6.26869 => "1lk3AH_H.int"
6.2607 => "1kshAB_A.int"
6.2601 => "1dvrAB_A.int"
6.25971 => "3k1iBC_B.int"
6.25893 => "1xu1BS_B.int"
6.25727 => "3q87AB_A.int"
6.25604 => "3fiwAB_A.int"
6.15399 => "1spbPS_P.int"
6.03797 => "3b94AB_B.int"
6.02542 => "1t2pAC_A.int"
6.00838 => "1zq7AD_A.int"
4.52721 => "3f5nBC_C.int"
4.52273 => "3eivAD_A.int"
4.51282 => "3h35BC_C.int"
4.5 => "1mkoAC_A.int"
4.36667 => "2ahmCH_H.int"
4.20776 => "2bcoAB_B.int"
4.19209 => "1g28AC_C.int"

Table A.6: LISA's top 25 results in descending order for ("1F5O", "C") with similarity scores.

60295.6 => "3lhbCH_C.int"
1073.32 => "3lhbCH_H.int"
581.088 => "3lhbBC_C.int"
16.6809 => "2fm1AC_A.int"
6.81977 => "1it2AB_A.int"
6.45 => "3lhbBC_B.int"
6.43182 => "2ybfAB_B.int"
6.2698 => "2go7BC_B.int"
6.2677 => "3nwvBD_B.int"
6.01786 => "3kdoBF_B.int"
5.82594 => "1r4aFH_F.int"
5.03514 => "3l8hAB_A.int"
4.52299 => "2qkuAC_A.int"
4.52062 => "1ybgBD_B.int"
4.31667 => "3e05BG_B.int"
4.24359 => "1be3AI_A.int"
4.17698 => "3q8nCD_C.int"
4.13333 => "3qkuAB_A.int"
4.125 => "1w72AD_A.int"
4.02817 => "2nzjAB_A.int"
4.02198 => "2aucCD_C.int"
4.01786 => "3nwvBD_D.int"
4.01538 => "3htaAC_A.int"
4.00409 => "3fsnAB_B.int"
3.69569 => "2pmyAB_B.int"

Table A.7: LISA's top 25 results in descending order for ("2WW9", "N") with similarity scores.

99.0 => "2ww9AN_N.int"
9.10526 => "1qeyCD_C.int"
8.02083 => "3fxdAB_A.int"
6.34524 => "1yw6AB_A.int"
6.27469 => "3dgcLS_L.int"
6.26786 => "3lfkAC_A.int"
6.2613 => "3t6kAB_B.int"
6.26036 => "2pjuAB_A.int"
6.25877 => "2c1wBC_C.int"
6.25746 => "2vozAB_A.int"
6.25613 => "1wwrAB_A.int"
6.25514 => "2c7yAB_A.int"
6.18382 => "3tfwAB_A.int"
6.11111 => "2yggAB_A.int"
6.01141 => "3dpiAB_A.int"
6.00691 => "2a72AB_B.int"
4.53883 => "2c9nYZ_Z.int"
4.51639 => "2vm8CD_C.int"
4.51587 => "1slqEF_F.int"
4.50835 => "3brqAB_B.int"
4.26667 => "2o1tHJ_H.int"
4.20667 => "2nnnAD_A.int"
4.2037 => "3rv5CD_C.int"
4.19322 => "2bw3AB_B.int"
4.19048 => "3r8tJQ_Q.int"

Table A.8: LISA's top 25 results in descending order for ("2XPJ", "C") with similarity scores.

14.1116 => "1yq2AF_A.int"
14.0795 => "2cjtBD_D.int"
12.9075 => "3dmmAC_C.int"
10.6065 => "3p0cAB_A.int"
9.52322 => "2dsoAE_E.int"
9.49732 => "1z0aAC_C.int"
8.51211 => "1nugAB_A.int"
8.10537 => "3nr4AC_C.int"
7.60694 => "2pgcBD_B.int"
6.52653 => "3n9xAB_B.int"
6.44202 => "1zvpAB_B.int"
6.33989 => "2cjtBD_B.int"
6.2613 => "1xmmBD_D.int"
6.25806 => "2y81AB_A.int"
6.16128 => "1y14BD_D.int"
6.06122 => "1zvsAD_A.int"
6.05769 => "2ov7BC_B.int"
6.04688 => "3t1vAB_B.int"
5.85175 => "3b5hAD_A.int"
5.81126 => "3cwqAB_A.int"
5.61725 => "1p1hBD_D.int"
5.35977 => "1gmyAB_A.int"
5.13657 => "3eh8AD_D.int"
5.04989 => "2q3xAB_B.int"
4.96757 => "1u1bAB_A.int"

Table A.9: LISA's top 25 results in descending order for ("2LMP", "N") with similarity scores.

0.583333 => "1t8lAD_A.int"
0.5625 => "3t1nAB_A.int"
0.538462 => "3geqAB_B.int"
0.533333 => "1j9kAB_B.int"
0.529412 => "2p85AD_D.int"
0.522727 => "2h42BC_B.int"
0.518182 => "2c4mBC_B.int"
0.517241 => "3cunAB_A.int"
0.515873 => "1xlqAB_A.int"
0.515625 => "1xlqAB_B.int"
0.515385 => "1sf8CD_D.int"
0.514706 => "1xv9AC_A.int"
0.513514 => "1uf2BF_F.int"
0.513333 => "1lqgAB_B.int"
0.512821 => "3hp3DH_D.int"
0.512658 => "3larAF_A.int"
0.511905 => "2vspAD_A.int"
0.51087 => "2vspAD_D.int"
0.510753 => "3a1yCG_C.int"
0.510526 => "3p1dAB_B.int"
0.509709 => "3k2bGQ_Q.int"
0.509615 => "1cbkAB_B.int"
0.508475 => "2v6sAB_A.int"
0.508264 => "3sp4AB_B.int"
0.50813 => "2hlzBD_D.int"

Table A.10: LISA's top 25 results in descending order for ("2LMP", "O") with similarity scores.

0.7 => "1qh2AB_A.int"
0.642857 => "2waxCD_D.int"
0.5625 => "3ik9AF_A.int"
0.555556 => "3k2aAB_B.int"
0.547619 => "2olkCD_D.int"
0.541667 => "3meqAB_B.int"
0.54 => "2e0iAC_C.int"
0.538462 => "3ooxAB_A.int"
0.537037 => "3fnmBD_D.int"
0.525 => "1vf5OR_O.int"
0.52439 => "3cueDQ_Q.int"
0.523256 => "2xf8EH_E.int"
0.521739 => "3k99AD_A.int"
0.52 => "3kymDK_D.int"
0.517544 => "1vspRW_R.int"
0.516393 => "3ezzAE_E.int"
0.516129 => "1o19DW_D.int"
0.515385 => "3k81AD_A.int"
0.514925 => "3hp3FJ_J.int"
0.514706 => "1szhAB_A.int"
0.514286 => "1xszAB_B.int"
0.513699 => "3hklAB_B.int"
0.513514 => "3gnlAB_B.int"
0.513158 => "3fin4G_G.int"
0.512821 => "2ialBD_D.int"

Table A.11: LISA's top 25 results in descending order for ("2PWH", "A") with similarity scores.

22.1429 => "1dyoAB_A.int"
9.90417 => "1i5cAB_A.int"
6.59489 => "2fh1AB_A.int"
6.38333 => "1v1hCF_F.int"
6.31667 => "1w87AB_B.int"
5.1 => "3h52BD_D.int"
5.05038 => "2wzjCD_C.int"
4.78968 => "3l2eAD_D.int"
4.70833 => "1iruFM_M.int"
4.59091 => "2qrwBE_E.int"
4.55593 => "2fqpAB_A.int"
4.54494 => "2xk5AB_A.int"
4.5339 => "3u0jAB_A.int"
4.46714 => "3o44KL_L.int"
4.31987 => "1g1kAB_A.int"
4.2381 => "3h9jDG_D.int"
4.23333 => "2c47CD_C.int"
4.23214 => "1g2yAC_C.int"
4.2193 => "2vlvAB_B.int"
4.21429 => "3mh4AB_B.int"
4.21354 => "3qu4AF_F.int"
4.21144 => "3ob9CD_D.int"
4.20196 => "1aznAC_A.int"
4.19728 => "1fkiAB_A.int"
4.12576 => "3h43EF_E.int"

Table A.12: LISA's top 25 results in descending order for ("1UIR", "A") with similarity scores.

30.3811 => "1qwtAB_A.int"
11.6526 => "2q1fAB_A.int"
9.26997 => "2a22AB_A.int"
8.2 => "2fp7AB_A.int"
7.43097 => "1gy7CD_D.int"
7.2463 => "3oj4DE_D.int"
7.09963 => "3hwxAI_I.int"
6.14458 => "3imlBD_B.int"
5.85175 => "1ow0BD_D.int"
5.57432 => "2xfbAD_D.int"
5.426 => "2xm3EF_E.int"
5.33986 => "2ap6AB_B.int"
5.06644 => "2c21CD_C.int"
4.98621 => "3r8bDG_D.int"
4.81444 => "2hzvFH_H.int"
4.61075 => "1ejhCD_C.int"
4.52614 => "2cnmBE_B.int"
4.51671 => "3robCD_C.int"
4.34128 => "1nycAB_A.int"
4.22789 => "2vx3BD_D.int"
4.22222 => "3robAD_A.int"
4.22024 => "3owuAC_A.int"
4.21839 => "1vdmFJ_J.int"
4.20721 => "2f6aCD_D.int"
4.20488 => "3f8xCD_D.int"

Table A.13: LISA's top 25 results in descending order for ("1YSF", "A") with similarity scores.

9.00833 => "1ll4AB_B.int"
8.11385 => "3gwrAB_B.int"
6.28226 => "1ir2DV_V.int"
6.27105 => "2bhmCD_C.int"
6.26481 => "3a17AD_D.int"
6.26449 => "3l2zAB_A.int"
6.26389 => "3agyAC_A.int"
6.26307 => "3dkwFG_F.int"
6.25475 => "1iq6AB_B.int"
6.25434 => "2hcrAB_A.int"
6.01007 => "2xvoAC_A.int"
6.00852 => "3demAB_B.int"
6.00748 => "3izvJV_J.int"
5.33909 => "1efvAB_A.int"
4.20196 => "2or8AB_A.int"
4.18707 => "1j6rAB_B.int"
4.1864 => "1nkqBF_B.int"
4.17971 => "2pr6AB_A.int"
4.17867 => "2ayoAB_B.int"
4.17843 => "1wmsAB_A.int"
4.1777 => "1uw4CD_C.int"
4.17407 => "3ew2FG_G.int"
4.17361 => "2vucAB_A.int"
4.09508 => "2i99AB_B.int"
4.03279 => "3qitBD_D.int"

Table A.14: LISA's top 25 results in descending order for ("3RYF", "E") with similarity scores.

13.52 => "1hqrCD_D.int"
9.07485 => "1oypAB_A.int"
8.03947 => "2whtBC_C.int"
6.48333 => "2y22EF_E.int"
6.44386 => "1xhxAB_A.int"
6.38333 => "3f9kBK_K.int"
6.26754 => "3tqmAC_C.int"
6.2655 => "2ahcAD_D.int"
6.26439 => "3ciqIJ_I.int"
6.2583 => "2yi2JQ_J.int"
6.25641 => "2iabAB_B.int"
6.25602 => "2f4pAB_A.int"
6.25252 => "3m1cAB_A.int"
6.15184 => "1oypAB_B.int"
6.07895 => "3irhAC_A.int"
6.05882 => "3ecrAB_B.int"
6.00968 => "2hqyAB_A.int"
6.00898 => "1jk9CD_C.int"
5.35125 => "3mq7EK_E.int"
5.19414 => "2v0rAB_A.int"
5.04207 => "2ib0AB_B.int"
4.91667 => "3r2cAK_A.int"
4.83961 => "3gdzAD_A.int"
4.78876 => "3k7dAB_B.int"
4.73511 => "3eo8AD_D.int"

Table A.15: LISA's top 25 results in descending order for ("1UIR", "B") with similarity scores.

26.0751 => "2byuAI_A.int"
17.8634 => "2p04AB_A.int"
13.2083 => "1zosCF_C.int"
10.2276 => "3iwuBF_B.int"
10.141 => "3itwAB_B.int"
10.1039 => "1om9AB_B.int"
9.95625 => "1tlyAB_B.int"
9.03448 => "1u7zBC_C.int"
8.23889 => "2wadAB_B.int"
8.13759 => "2ql6GH_G.int"
8.02649 => "2gtpAD_A.int"
7.71904 => "2pquCD_C.int"
6.73504 => "1lhrAB_A.int"
6.43182 => "1htlCD_D.int"
6.36111 => "1vf5PQ_P.int"
6.32143 => "2guzAC_A.int"
6.29545 => "1sg2BC_C.int"
6.28846 => "2imjBC_C.int"
6.27597 => "3i7vAB_A.int"
6.27222 => "1iqpCD_C.int"
6.2702 => "1kvoEF_F.int"
5.83874 => "3oirAB_B.int"
5.63846 => "3em0AB_B.int"
5.61178 => "2q9uAB_A.int"
5.58905 => "2o62AB_B.int"

Table A.16: LISA's top 25 results in descending order for ("2PWH", "B") with similarity scores.

17.7461 => "2wzjCD_C.int"
10.4156 => "3bimBI_B.int"
8.09375 => "1i5cAB_A.int"
7.59995 => "1iwaFI_F.int"
6.81568 => "3qq2BC_C.int"
6.60246 => "2fqpAB_A.int"
6.27353 => "1aznAC_A.int"
5.86437 => "1fvjAB_A.int"
5.21693 => "3pu2CF_F.int"
5.125 => "2wttJK_J.int"
4.95 => "2ahmGH_G.int"
4.90714 => "3o01AB_B.int"
4.66667 => "3u5eBO_O.int"
4.63393 => "1oypAF_F.int"
4.6 => "2a41AC_C.int"
4.55128 => "2ialBD_D.int"
4.54622 => "3p8cAF_F.int"
4.53452 => "2i52BF_B.int"
4.2971 => "1qnzHP_P.int"
4.25833 => "2r2vBD_D.int"
4.23333 => "2c47CD_C.int"
4.22222 => "2a45KL_K.int"
4.21014 => "3er6AE_A.int"
4.20155 => "1g8kCH_H.int"
4.13333 => "1v1hCF_F.int"

Table A.17: LISA's top 25 results in descending order for ("3JZT", "F") with similarity scores.

9.01053 => "1rf5BC_B.int"
8.43678 => "3lk6AD_D.int"
8.18475 => "2ab6BD_D.int"
8.02174 => "3ffuAB_B.int"
6.48475 => "3lk6AD_A.int"
6.27198 => "2y6eEF_F.int"
6.2584 => "2wmmAB_A.int"
5.02976 => "2qyaAB_B.int"
4.56667 => "3laiAC_C.int"
4.54444 => "2a5vAD_D.int"
4.53333 => "2csxAB_A.int"
4.51878 => "1po9AB_A.int"
4.22327 => "2omoCD_D.int"
4.20238 => "1it3AD_A.int"
4.19858 => "2p2lBC_B.int"
4.19792 => "2vatAK_A.int"
4.18939 => "2oncBC_C.int"
4.18889 => "3hp3GI_G.int"
4.18421 => "2gr8AE_A.int"
4.1775 => "2z2oAD_D.int"
4.17677 => "1s28CD_C.int"
4.17409 => "1yukAB_A.int"
4.09785 => "2deoAB_A.int"
4.04651 => "2jfdBC_C.int"
4.02469 => "2xnkAC_C.int"