AN ALGORITHM FOR FINDING A BETTER TM-SCORE

LI SHUAI GUO

MASTER OF COMPUTER SCIENCE

FACULTY OF INFORMATION AND
COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN

SEPTEMBER 2018

AN ALGORITHM FOR FINDING A BETTER TM-SCORE

By

LI SHUAI GUO

A dissertation submitted to the Department of Computer Science,
Faculty of Information and Communication Technology,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Computer Science
September 2018

ABSTRACT

AN ALGORITHM FOR FINDING A BETTER TM-SCORE

There are many scoring functions have been proposed to evaluate the similarity between protein structure models. Among these, a popular measure is the template modeling score (TM-score), introduced by Zhang and Skolnick. At this moment, the TM-score is calculated through a heuristic algorithm with no accuracy guarantee. In this paper, we propose an algorithm which computes more accurate TM-score, through the use of the very fast Kabsch-which is commonly used to compute the Root Mean Square Deviation (RMSD). Our algorithm the first obtain an approximation for superposition of the protein model that optimizes the TM-score (for example, through Opt (GDT). Then, iteratively refines this superposition through the rotation axes discovered using the Kabsch algorithms. The algorithm is implemented in C++ into a tool that runs in a time comparable to Zhang and Skolnick's TM-score software, but consistently produces TM-score that are more accurate.

ACKNOWLEDGEMENT

I am deeply grateful to my supervisor, who has been supper talent and he always is bringing out the crucial points in my papers. I very appreciate for giving me help. Dr. Ng Yen Kaow for his endless patience, motivation, guidance, and support that has helped me throughout my research and to finish this dissertation. I would like to express my special gratitude to my co-supervisor, Dr. Goh Yong Kheng who has been a delightful person to work with. He had given me a lot of ideas and knowledge throughout the research. I would also like to thank Universiti Tunku Abdul Rahman (UTAR) for the financial and facilities support.

I would also like to thank the lecturers and staff in UTAR who were involved in this research. The knowledge and experiences they have shared with me are priceless. Without their passionate guidance, this research may not have been successfully completed. I would also like to thank my friends for accepting nothing less than excellence from me.

Last but not the least, I would like to thank my family especially my parents for providing me with lots of support and continuous encouragement throughout my years of study. Their advice is always the best and practical when making a life decision.

APPROVAL SHEET

This dissertation/thesis entitled "**AN ALGORITHM FOR FINDING A BETTER TM-SCORE**" was prepared by LI SHUAI GUO and submitted as partial fulfillment of the degree of Master of Computer Science at Universiti Tunku Abdul Rahman.

Approved by:

_____

(Prof. Dr. Ng Ken Kaow) Date: …….

Supervisor

Department of Computer Science

Faculty of Information and Communication Technology

Universiti Tunku Abdul Rahman

_____

(Prof. Dr. Goh Yong Kheng) Date: ……..

Co-supervisor

Department of Computer Science

Faculty of Information and Communication Technology

Universiti Tunku Abdul Rahman

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: _____

SUBMISSION OF DISSERTATION

It is hereby certified that Li Shuai Guo (ID No: 13ACM06823) has completed this dissertation entitled "AN ALGORITHM FOR FINDING A BETTER TM-SCORE" under the supervision of Dr. Ng Yen Kaow (Supervisor) from the Department Computer Science, Faculty of Information and Communication Technology, and Dr. Goh Yong Kheng (Co-supervisor) form the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my dissertation in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and the public.

Yours truly,

_____

(Li Shuai Guo)

DECLARATION

I, Li Shuai Guo, hereby declare that the dissertation is based on my original work except for quotation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

_____

(LI SHUAI GUO)

Date _____

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

A DNA, or *deoxyribonucleic acid*, is a molecule that consists of a very long chain of nucleotides (Alberts *et al.*, 2014). A nucleotide consists of a sugar (deoxyribose) and one of four bases: Cytosine (C), Thymine (T), Adenine (A), and Guanine (G). The DNA of an organism encodes the genetic information needed to carry out the biological processes of the organism.

DNA works by copying a small portion of its genetic codes into shorter molecules called RNAs, or *ribonucleic acids*. The RNA transcribed by a segment of DNA is identical to the DNA except that Thymine is replaced by Uracil (U). An RNA functions by copying itself into its corresponding amino acids sequence. The amino acid sequence, in turn, folds into a stable three-dimensional structure called a *protein*, driven by the physical forces of its constituent amino acids. This mechanism of how DNA produces proteins is known as the Central Dogma of Molecular Biology (Figure 1.1).

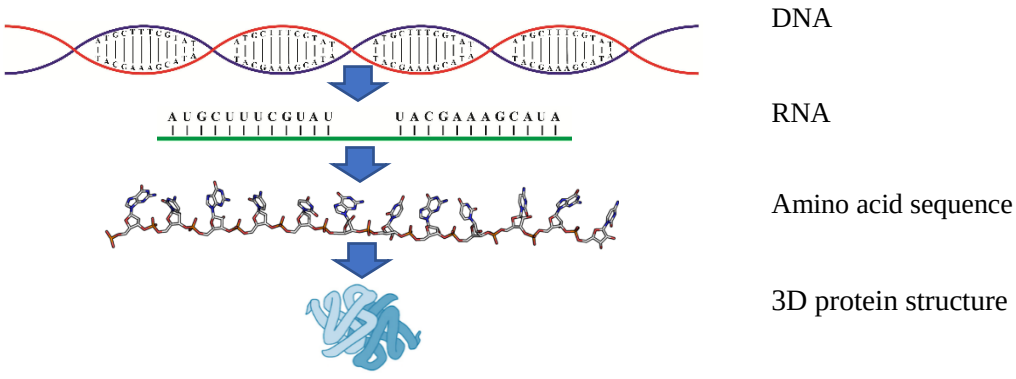

Figure 1.1: Central Dogma of Molecular Biology

The function of a protein is directly dependent on its three-dimensional structure (Alberts *et al.*, 2014). Hence, one can identify the function of an unknown protein by comparing its structure to those from proteins of known functions. For this reason, many researches have focused on the task of comparing between two protein structures.

The comparison between protein structures is also an important task in protein structure prediction (Kufareva and Abagyan, 2012), where one infers the structure which an amino acid sequence fold into. They serve at least two important functions. First, structure comparison is a subroutine when we need to select a representative structure from a collection, which is a task that is often required in many protein structure prediction methods. Second, they are needed when we want to evaluate the success of a protein prediction method by comparing the output structure of the method against the known target structure.

1.2 Problem Statement

Using a similarity measure is the common approach to compare two protein structures. Such a similarity measure would map each amino acid in one of the structures to a corresponding amino acid in the other structure. The distances between corresponding amino acids are then collected and used to produce a final score.

A few measures of similarity are routinely used in protein structure comparison, they are the Root Mean Square Deviation (RMSD) (Kabsch, 1976), Local-Global Alignment (LGA) (Zemla *et al.*, 1999; Zemla, 2003), MaxSub

(Siew *et al.*, 2000), and Template Modeling Score (TM-score) (Zhang and Skolnick, 2004).

The computation of all of these measures are complicated by the fact that different superpositions of the structures would result in different sets of distances between the amino acids (Figure 1.2). All the similarity measures require us to consider every possible superposition of the two structures in their computations.

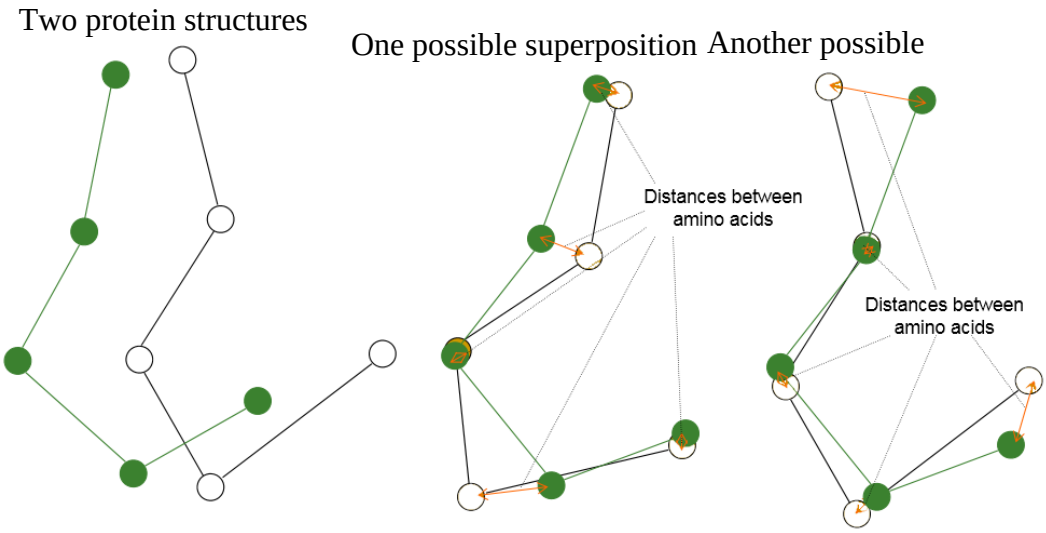

Figure 1.2: Distances between amino acids change with superposition

There are some well-known shortcomings with some of these measures. For instance, there are at least two shortcomings with the RMSD, which is defined as the sum of the squared values of all the inter-amino acid distances. First, the value of the measure is hard to interpret across different situations. For example, a value of RMSD=3Å (Angstrom) may indicate that the structures are very similar in a case with long structures, but dissimilar in another case with short structures. Ideally, a measure that can be interpreted similarly across

different situations should have values that are normalized to lie between a fixed range, such [-1, 1] or [0, 1]. However, the RMSD has a range of (0, +∞).

Second, since the distances are squared in the RMSD, the measure places a tougher penalty on larger inter-amino acid distances. For example, in Figure 1.3, the two structures in case A are identical except for a pair which differs by a relatively large distance. However, this comparison would result in a far larger RMSD than the two structures in case B.

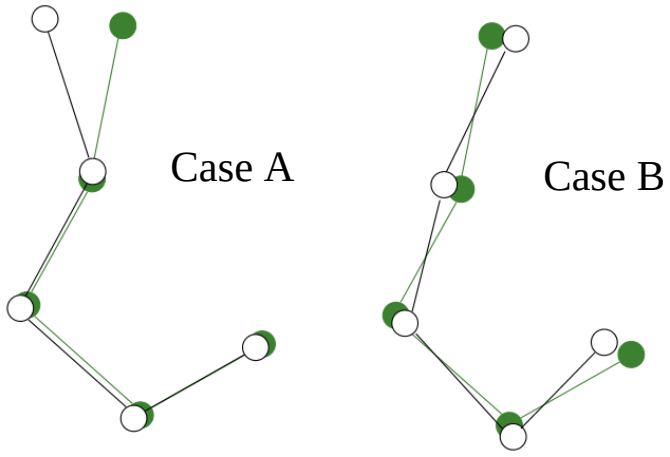

Figure 1.3: Two cases of structure comparison

These shortcomings in the RMSD has prompted the creation of more sophisticated similarity measures. MaxSub discovers the largest subset of amino acids that match well and uses that subset to produce a normalized score. In LGA, the distance measure is split into a "local" component, called Longest Continuous Segment (LCS), and a "global" component, called Global Distance Test (GDT) components. Finally, given two protein structures $A = (a_1, a_2, \ldots, a_n)$ and $B =$

$(b_1, b_2, \ldots, b_n)$, where a_i and b_i respectively represents the coordinates of the amino acids in *A* and *B*, the TM-score between *A* and *B* is defined as

$$\text{TM-Score}(A, B) = \frac{1}{n} \max_{R,T} \sum_{i=1}^{n} \frac{1}{1+\left(\frac{\|Ra_i - b_i - T\|}{d_0}\right)^2}$$

where d_0 is a normalization factor given as $d_0 = 1.24(n-15)^{1/3} - 1.8$. Through this formulation, it is easy to see that the TM-score has a range between 0 and 1, with very similar structures scoring close to 1 and dissimilar structures scoring close to 0. This naturally avoids the first problem faced by the RMSD. The TM-score also does not penalize far away amino acids pairs; such pairs would simply contribute little towards the score.

These measures have been used in the Critical Assessment of protein Structure Prediction (CASP), a competition held biennially to evaluate the success of protein structure prediction methods. They are used mainly to evaluate how close the outputs of the methods are to the actual protein structures. While they do not suffer from the problems faced by the RMSD, unlike the RMSD there are no exact algorithms for their computation.

1.3 Motivation

This thesis studies the TM-score, which is favored by the CASP community. The computational complexity for finding the TM-score is unknown. At this moment, the TM-score is calculated through a heuristic method. There is no known algorithm for computing TM-score with a theoretical basis. In particular, there is currently no computation with a theoretical guarantee on the correctness of the score it obtains.

This thesis aims to improve on the current heuristic algorithm for computing the TM-score.

The current TM-score computation involves two heuristic steps. At each step, the algorithm only optimizes the TM-score with respect to only a segment of the structure. It is not known how this "local" aspect of the optimization would impact the result of the heuristic algorithm.

It is worth investigating if by changing these steps to optimize the TM-score in more "global" sense, more accurate TM-scores can be obtained.

1.4 Objectives

The following are the objectives of this research:

- Propose an improved method for the computation of the TM-score, in particular:
    - The method is to optimize global aspects of the TM-score at each step as opposed to the currently available algorithm.
    - The method should have similar runtime as the currently available algorithm.
- Create a fast and usable software tool based on the algorithm. In particular:
    - The program is not to depend on external libraries, so that it may be compiled on as many platforms as possible and distributed as a standalone tool for use by researchers.
- Perform extensive comparison on the proposed algorithm against the currently available tool for finding TM-score.

- The comparison is to be performed with a comprehensive set of structures that is relevant to the field of interest, that is, protein structure prediction.
- The TM-scores computed from the algorithm are to be compared against those computed from the currently available tool.
- The times taken for the tool are to be benchmarked against the times taken by the currently available tool.

1.5 Scope

In this research, a new algorithm for computing the TM-score was proposed. Like the currently available algorithm, the new algorithm is heuristic and iterative in nature. However, it optimizes global aspects of the TM-score during every iteration of its computation.

The algorithm was implemented as a standalone C++ program which requires no external library (other than ANSI libraries).

Finally, the binary compiled from the program was tested using a comprehensive set of data from the CASP test set. The performance of the tool was compared against the currently available tool for TM-score. Using this comparison, the tool is shown to be more accurate than the currently available tool, while running in comparable time.

1.6 Research Contributions

The major contributions of this research are as follows:

- An improved heuristic and iterative method for the computation of the TM-score, which is able to optimize global aspects of the TM-score at each step as opposed to the currently available algorithm.
- An auxiliary branch and bound method to speed up the proposed algorithm.
- A fast and usable alternative software tool (compiled on multiple OS platforms) for computing the TM-score. The tool can be used as a replacement to the current tool for finding TM-score, or as a mean to verify the output of the current tool.

1.7 Organization of dissertation

Chapter 2 presents a literature review of the existing results in molecular biology that led to the present problem, as well as more in-depth discussions of these algorithms. The new algorithm proposed in this thesis is explained in Chapter 3. Chapter 4 shows the experimental results and compares the results of the proposed approach with those from the currently available method. Chapter 5 gives some discussions and concludes the thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 Bio-sequences

Large-scale sequencing of the DNA became a possibility since the Sanger chain sequencing technique was developed in 1977 (Sanger and Coulson, 1977). In 1998, a method known as pyrosequencing further improved sequencing speed (Ronaghi et al., 1998). In the subsequent years, a few sequencing techniques, now commonly referred to as next-generation sequencing (NGS) together with pyrosequencing, were invented. Their availability has made DNA information easily available in biological and medical researches. This has yielded a very large collection of DNA sequences for analysis.

2.2 Sequence comparison

Given the huge database of sequence, fast algorithms for comparing sequences became important in the past decades. In particular, they are needed for the following tasks:

- Identifying the contributor of an unknown sequence

  Given an unknown sequence, we can match the sequence to a database of sequences where the contributors are known, in order to either identify the species, or even the exact organism which the sequence belongs to.

- Predicting the functions of an unknown sequence

The genetic sequence of an organism determines the organism's physical traits, and similar sequences often lead to the same traits. Hence, given an unknown sequence, we can predict its function by finding sequences with known functions that are similar to it. The functions of those sequences are then likely to be the functions of the unknown sequence.

- Finding the genes within a sequence

  Given a database of genes and an unknown sequence, we can find which genes in the database exist in the sequence.

In order to biological sequences, a score is typically defined to express the difference between two sequences and then an algorithm is then designed to minimize this score. Examples of such scores are the Hamming distance and the edit distance.

The first well-known algorithm, the Needleman-Wunsch algorithm, for comparing sequences was proposed in 1970 (Needleman and Wunsch, 1970). This was followed by the Smith-Waterman algorithm (Smith and Waterman, 1981).

2.3 Protein structure prediction

According to the Central Dogma of Molecular Biology, DNA works by transcribing its sequences into RNA, and subsequently, into protein structures, which then carries out biological functions of the organism. In a sense, our study of DNA sequences is motivated by our desire to understand these protein structures.

The portions in a DNA sequence which are involved in these transcriptions are known as genes. Genes must be transcribed into proteins in to perform their functions. Furthermore, the function of a protein relies mostly on its structure. Evolutionarily, protein structures are 3 to 10 times better conserved than their sequences. Hence, to predict the function of a genetic sequence, some researchers first infer the protein which it is transcribed into and then compare the structure of that protein with protein structures of known functions. The similarity in the structures would then give a better prediction for the function of the original sequence.

Due to the mechanism of genetic folding, it is possible to predict the protein structure which a gene encodes. This has resulted in the study of protein structure prediction in the last two decades (Dorn *et al.*, 2014).

Given a gene sequence, there are four levels of structures in which protein structure prediction can be performed.

(1) Primary structure: this predicts the linear arrangement of amino acids in a protein and the location covalent linkages such as disulfide bonds between amino acids.
(2) Secondary structure: this predicts the areas of folding or coiling within a protein; examples include alpha helices and pleated sheets, which are stabilized by hydrogen bonding.
(3) Tertiary structure: this predicts the final three-dimensional structure of a protein, which results from a large number of non-covalent interactions between amino acids.
(4) Quaternary structure: this predicts the non-covalent interactions that bind multiple polypeptides into a single, large protein.

The primary structure of a protein can be readily deduced from the nucleotide sequence of the corresponding messenger RNA, based on primary structure. Many features of secondary structures can be predicted with the aid of computer programs. However, predicting protein tertiary and quaternary structures remains very tough problems.

The comparison of protein structures is a recurring problem in protein structure prediction. There are two main ways in which protein structures are compared:

- Structural alignment
- Model comparison

In both types of comparison, two structures are given in the problem statement, typically as. A protein structure *A* consists of an ordered set of n points in three-dimension, denoted $(a_1, a_2, \dots, a_n)$. Each point a_i gives the coordinates of the C_a atom in the i-th amino acid. A structure *B* consisted of m points, denoted $(b_1, b_2, \dots, b_m)$. There are many ways to formulate both the structural alignment problem and the model comparison problem. In all the formulations, a scoring function that measures how similar (or dissimilar) the two structures are is defined, and the problem is to find a way to compute the scoring function effectively. One difference between structural alignment and model comparison is that: one is to first find an order-preserving one-one mapping between the points in *A* and *B* in structural alignment, whereas such a mapping is provided in model comparison.

This thesis is concerned with the latter, model comparison.

2.4 Model comparison

In model comparison, one is given two protein structures, $A=(a_1, a_2, \ldots, a_n)$ and $B=(b_1, b_2, \ldots, b_n)$, and is required to determine how similar the two structures are. There are many different formulations to the problem depending on the scoring function. Several scoring functions have been proposed for the purpose of protein structure prediction, such as Root Mean Square Deviation (RMSD) (Kabsch, 1976), Local-Global Alignment (LGA) (Zemla *et al.*, 1999; Zemla, 2003), MaxSub (Siew *et al.*, 2000), and Template Modeling Score (TM-score) (Zhang and Skolnick, 2004).

Model comparison serves several purposes in protein structure prediction, among which the following two are most prominent:

1. For the evaluation of the predicted protein structure against the known structure. The predicted structure is known as a "*model*" structure while the known structure is called the "*native*" structure in the literature.
2. For the selection of a consensus structure out of a collection of similar structures generated typically using some sampling method such as Gibbs Sampling.

The root means square deviation (RMSD) is one of the earliest structural comparison measure proposed (Nishikawa *et al.*, 1972; Rao and Rossmann, 1973), as well as the best studied. For two structures $A=(a_1, a_2, \ldots, a_n)$ and $B=(b_1, b_2, \ldots, b_n)$, the RMSD is defined as

$$\mathrm{RMSD}(A,B) = \min_{R\in\mathcal{R},T\in\mathcal{T}} \sqrt{\frac{\sum_{i=1}^{n}\|Ra_i - b_i - T\|^2}{n}},$$

where T is some translation in the space of all translations $\mathcal{T}$, and R is some rotation in the space of all rotations $\mathcal{R}$. Kabsch first gave an algorithm which computes the RMSD in linear time (Kabsch, 1976), as follows:

Input: Protein structures *A* and *B*.

(1) Translate A and B with a translation T' which result in their centroids to coincide.

(2) Find the 3x3 matrix $C = BA^T$. (denotes A^T the transpose of A.)

(3) Find the Single Value Decomposition (SVD) of *C*. That is, find U, V and diagonal S such that $C = U^T SV$.

(4) Output the RMSD as $\frac{1}{n}\sum_{i=1}^{n} {p_i}^2 + {q_i}^2 - 2(l_1 + l_2 + l_3)$, where l_1, l_2, and l_3 are the singular values in *S*.

(The corresponding rotation $R = VU^T$ and translation $T = T'$ for this RMSD value.)

Table 2.1: Kabsch Algorithm

Due to its low runtime complexity, the RMSD has come in very convenient for the comparison of structures. However, it suffers from a few drawbacks. First, as mentioned in Chapter 1, an RMSD value of 3Å (Angstrom) may indicate high similarity between two structures of a few hundred points but would be

considered very dissimilar for two structures of only a few points. More precisely, in order to a measure to have universal interpretation across different scenarios, its range is typically normalized to within some interval of values, e.g. [0, 1] or [-1, 1]. However, the range of RMSD is between (0, +∞).

Second, since the distances are squared in the RMSD, the measure places a tougher penalty on larger inter-amino acid distances, as demonstrated in Figure of Chapter 1.

These shortcomings have resulted in the proposal of other similarity measures, such as the GDT, MaxSub, and TM-score.

2.5 GDT

To avoid the problems faced by the RMSD, Zemla *et al.* (1999) introduced a measure called the Local-Global Alignment (LGA). LGA consists of a "local" component, called the Longest Continuous Segment (LCS), and a "global" component, called the Global Distance Test (GDT). The latter, GDT, has received widespread adoption in the community.

GDT is defined on a sub-problem known as *d*-LCP, which aims to find the largest common point sets under approximate congruence for the given distance threshold d. More precisely, given two structures $A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_n)$ and threshold d, *d*-LCP aims to find the largest set *M* of pairs of (a_i, b_i) which fulfills

$$(\forall\ (a_i, b_i) \in M)[\|Ra_i - b_i - T\| \leq d].$$

for some $T \in \mathcal{T}$ and $R \in \mathcal{R}$.

The GDT is then, computed as a composite of the four d-LCP scores where d is set to 1Å, 2Å, 4Å and 8Å.

Unlike the RMSD which places a heavy penalty on unmatchable amino acids, the GDT simply discounts them.

While the d-LCP problem can be solved in $O(n^7)$ time (Li *et al.*, 2008), the high time complexity makes the algorithm impractical. Currently, GDT is computed through a heuristic algorithm.

Intuitively, the algorithm starts with an initial subset of amino acid pairs that can be superposed to within the threshold distance d. Then, it iteratively attempts to "grow" the set of amino acids. To do so, the RMSD is used as a subroutine. At every iteration, an RMSD is calculated to obtain an optimal superposition for the current set of matching amino acids; this superposition is then used to identify more matching amino acids.

This same strategy is used by many subsequent researchers (Siew *et al.*, 2000; Ortiz *et al.*, 2002; Kihara and Skolnick, 2003; Zhang and Skolnick, 2004).

The following shows this algorithm in detail:

Input: Protein structures *A*, *B*, and distance threshold d.

(1) For each pair of *3, 5, and 7* residue-long corresponding segments (A', B') from both structures,

  (1.1) Calculate an RMSD to obtain the corresponding superposition (R, T) which optimally superposes (A', B') for the RMSD.

  (1.2) Find the subset of amino acid pairs (A'', B'') within (A, B) where $[\|Ra_i - b_i - T\| \leq d]$.

  (1.3) Set A' to A'' and B' to B''.

  (1.4) Repeat (1.1) -(1.3) until there are no more changes to (R, T). (Hence, no more changes to (A', B').)

(2) Output the largest set (A', B') found with all different initial segments.

Table 2.2: Heuristic Algorithm for GDT

The algorithm achieves good results in general. However, since it is heuristic, there is no guarantee on whether the superposition found by the algorithm optimizes the number of matching amino acids found.

There is also worth noting that at every step of the computation, the optimal superposition is computed only on a subset of *A* and *B* and hence may miss out some "global" properties of the structures.

2.6 MaxSub

The MaxSub score is based on the GDT. Given two structures $A=(a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_n)$ and given threshold d, MaxSub aims to find a largest set *M* of pairs of (a_i, b_i) which fulfills

$$(\forall\ (a_i, b_i) \in M)[\|Ra_i - b_i - T\| \leq d]$$

Due to the similarity between MaxSub and GDT, the MaxSub score has a polynomial solution of very high order (Li *et al*., 2008).

At present, the most common way to compute the MaxSub score is through a heuristic algorithm similar to that used for computing the GDT.

2.7 TM-score

Given two protein structures $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$, where a_i and b_i respectively represents the coordinates of the amino acids in *A* and *B*, the TM-score between *A* and *B* is defined as

$$\text{TM-score}(A, B) = \frac{1}{n} \max_{R,T} \sum_{i=1}^{n} \frac{1}{1+\left(\frac{\|Ra_i - b_i - T\|}{d_0}\right)^2}$$

where d_0 is a normalization factor given as $d_0 = 1.24(n-15)^{1/3} - 1.8$. It is clear that the TM-score has values within (0,1].

Like the RMSD, the TM-score is defined through the distances between amino acids. However, TM-score is based on the inverse of the squared distances rather than the squared distances. Because of this, unlike the RMSD where a larger score indicates dissimilarity, a larger TM-score would indicate more similar structures.

Analytically solving the optimal superposition for the score in a straightforward fashion will require the solving of the roots of high order polynomials. Hence, interesting to know to what extent the TM-score can be computed accurately.

Currently, TM-score is computed through an algorithm that is identical to that for computing the GDT, except that when collecting the amino acids at each iteration, the criteria is changed to examine the condition $[\|Ra_i - b_i - T\| \leq d_0]$ rather than $[\|Ra_i - b_i - T\| \leq d]$. The entire algorithm is reproduced below for completeness.

Input: Protein structures *A* and *B*.

(1) For each pair of *3*, *5*, *and 7* residue-long corresponding segments (A', B') from both structures,

  (1.1) Calculate an RMSD to obtain the corresponding superposition (R, T) which optimally superposes (A', B') for the RMSD.

  (1.2) Find the subset of amino acid pairs (A'', B'') within (A, B) where $[\|Ra_i - b_i - T\| \leq d_0]$.

  (1.3) Set A' to A'' and B' to B''.

  (1.4) Repeat (1.1)-(1.3) until there is no more changes to (A', B').

(2) Output the optimal TM-score (A', B') found with all different initial segments.

Table 2.3: Heuristic Algorithm for TM-score

As with GDT, the algorithm achieves good results in general. However, since it is heuristic, there is no guarantee on whether the superposition found by the algorithm optimizes the number of matching amino acids found. This is particularly likely since the optimal superposition is computed only on a subset of *A* and *B* at each iteration.

On the other hand, since TM-score has gained popularity in the protein structure prediction community, its accuracy has become a matter of significant importance.

CHAPTER 3

METHODOLOGY

In this chapter, an algorithm which computes more accurate TM-score is developed. The algorithm follows the general framework of the iterative algorithm currently in use, but offers the following enhancements:

- Better iteration through gradient descent-like search,
- Instead of using only a subset of matching points, all the points in the two input structures are used in obtaining the superposition at each iteration.

As stated in the earlier chapter, given two protein structures $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$, the TM-score between *A* and *B* is

$$\text{TM-Score}(A, B) = \frac{1}{n} \max_{R,T} \sum_{i=1}^{n} \frac{1}{1 + \left(\frac{\|Ra_i - b_i - T\|}{d_0} \right)^2} \quad (1)$$

where d_0 is normalization factor given as $d_0 = 1.24(n - 15)^{1/3} - 1.8$ [11].

3.1 Difficulty of computing TM-score

It is unlikely that TM-score would yield an analytical closed-form solution. Consider the simplified case where the points in *A* and *B* have only components along the x-axis. In this case, no rotation is required, and Eqn. (1) becomes

$$\text{TM-Score}(A, B) = \frac{1}{n} \max_{R} \sum_{i=1}^{n} \frac{d_0^2}{d_0^2 + (a_i \cdot \vec{x} - b_i \cdot \vec{x} - x)^2} \quad (2)$$

where $\vec{x}$ is the unit vector along the x-axis and x is the displacement along the

x-axis. An attempt to obtain the optimal value for x by differentiating Eqn. (2) with respect to x and equating it with zero will result in a high order polynomial equation, for which the roots cannot be solved efficiently. Hence, even in the case of a single translation along a single axis, the problem of optimizing the TM-score is difficult.

3.2 A New algorithm for TM-score

Given structures *A* and *B*, the computation of TM-score (*A*, *B*) is the same as that of finding rotation *R* and matrix which maximizes

$$f(R,T) = \sum_{i=1}^{n} \frac{d_0^2}{d_0^2 + \|Ra_i - b_i - T\|^2}$$

Write the translation *T* as $\langle t_x, t_y, t_z \rangle$ (where $t_x, t_y, t_z \in \mathbb{R}$), and $(Ra_i - b_i)$ as $\langle r_{ix}, r_{iy}, r_{iz} \rangle$ (where $r_{ix}, r_{iy}, r_{iz} \in \mathbb{R}$). Then we can show that

$$f(R,T) = \sum_{i=1}^{n} \frac{d_0^2}{d_0^2 + \|Ra_i - b_i\|^2 + \|T\|^2 - 2\sum_{j=x,y,z} t_j r_{ij}}$$

Now, collect all the terms which do not depend on x into a new variable p_{ix}. That is,

$$p_{ix} = d_0^2 + \|Ra_i - b_i\|^2 + t_y^2 + t_z^2 - 2\sum_{j=y,z} t_j r_{ij}.$$

The expression can then be simplified into :

$$f(R,T) = \sum_{i=1}^{n} \frac{d_0^2}{t_x^2 - 2t_x r_{ix} + p_{ix}}$$

At this point, differentiating $f(R,T)$ with respect to t_x will result in

$$\frac{df(R,T)}{dt_x} = \sum_{i=1}^{n} \frac{-d_0^2(2t_x - 2r_{ix})}{(t_x^2 - 2t_x r_{ix} + p_{ix})^2} = \sum_{i=1}^{n} \frac{-d_0^2(2t_x - 2r_{ix})}{(d_0^2 + \|Ra_i - b_i - T\|^2)^2}$$

Denote $\frac{d_0^2}{\left(d_0^2+\|Ra_i-b_i-T\|^2\right)^2}$ as w_i. Then, the expression is simplified into

$$\frac{df(R,t)}{dt_x} = \sum_{i=1}^{n} -w_i(2t_x - 2r_{ix}).$$

The critical point of $f(R,T)$ is hence at

$$t_{x=} \frac{\sum_i w_i r_{ix}}{\sum_i w_i}. \tag{3}$$

Similarly, the optimal t_y and t_z can be shown to be at

$$t_y = \frac{\sum_i w_i r_{iy}}{\sum_i w_i}, \text{ and} \tag{4}$$

$$t_z = \frac{\sum_i w_i r_{iz}}{\sum_i w_i}, \text{ respectively.} \tag{5}$$

Hence, if an optimal rotation R is known, then an optimal T can be calculated from Eqn. (3)-(5), given that w_i is known.

This gives an opportunity for a method which iteratively improves on the TM-score, where at each iteration we

(1) Compute a semi-optimal R,

(2) Based on R, compute a semi-optimal T from the above equations,

(3) Repeat Step (1)-(2) until no further improvement can be obtained.

For the computation of T in Step (2), we assume R and T to be relatively small around convergence. In which case, we simply take $w_i = d_0^2/(d_0^2 + \|Ra_i - b_i - T\|^2)^2$, and compute t_x, t_y, and t_z using Eqn. (3)-(5).

We now discuss how to compute R in Step (1). This is achieved through optimizing the RMSD. The Kabsch algorithm is used for this purpose. We first relate our object our objective function f to the RMSD.

$$\begin{aligned} f(R,T) &= \sum_i \frac{d_0^2}{d_0^2 + \|Ra_i - b_i - T\|^2} \\ &= \sum_i w_i(d_0^2 + \|Ra_i - b_i - T\|^2) \\ &= \sum_i w_i d_0^2 + \sum_i w_i \|Ra_i - b_i - T\|^2 \end{aligned}$$

There are two terms in this expression of f. The first term depends only on w_i, while the second one resembles the objective function in the RMSD. The intuition given here is that, the RMSD is closely related to the current optimization problem. Hence, we would expect the rotation axis used to superimpose the structures in RMSD to be a good candidate for finding an optimal rotation for f. This rotation axis can be obtained from the Kabsch algorithm described in Table 2.1, without running Step (1).

Given this rotation axis, our algorithm will perform an exhaustive search on all the rotation angles about the axis to find an angle which optimizes f. This gives our final algorithm as follows.

Input: Protein structures *A* and *B*.

(1) Set initial $f_{\text{old}}(R,T)$ to 0, set $f_{\text{new}}(R,T)$ to 1 and initialize semi-optimal rotation *R* to $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$.

(2) Define a suitable accuracy threshold *t* (e. g. 0.0001), for stopping the iteration.

(3) While $|f_{\text{new}}(R,T) - f_{\text{old}}(R,T)| \geq t$, do

(3.1) Let $w_i = \frac{d_0^2}{\left(d_0^2 + \|Ra_i - b_i\|^2\right)^2}$.

(3.2) Let $T = \langle \frac{\sum_i w_i r_{ix}}{\sum_i w_i}, \frac{\sum_i w_i r_{iy}}{\sum_i w_i}, \frac{\sum_i w_i r_{iz}}{\sum_i w_i} \rangle$.

(3.3) Obtain R', the optimal rotation for obtaining the RMSD under translation *T*.

(3.4) Apply *T* and *R* on *A* and let $f_{\text{new}}(R,T) = f(R,T)$.

(4) Output $f_{\text{new}}(R,T)$ as the TM-score computed.

Table 3.1: Our Proposed Algorithm for TM-score

The computations for Step (3.1) -(3.2) is clear. For Step (3.3), the algorithm in Table 2.1 is performed with the centroid alignment Step (1) replaced with $T = \langle \frac{\sum_i w_i r_{ix}}{\sum_i w_i}, \frac{\sum_i w_i r_{iy}}{\sum_i w_i}, \frac{\sum_i w_i r_{iz}}{\sum_i w_i} \rangle$. Steps (3.5) -(3.6) are straightforward. Hence, we only need to discuss the computation of Step (3.4).

3.3 Finding an optimal θ

Recall that the TM-score is defined as

$$\text{TM-score}(A,B) = \frac{1}{n}\max_{R,T}\sum_{i=1}^{n}\frac{1}{1+\left(\frac{\|Ra_i - b_i - T\|}{d_0}\right)^2}$$

That is, it is the sum of n terms of the form $\frac{1}{1+\left(\frac{\|Ra_i - b_i - T\|}{d_0}\right)^2}$. Each term can be considered the contribution of the i-th amino acid pairs (i.e. a_i and b_i) towards the TM-score under a given superposition. We want to study how each of these individual contributions changes according to the rotation angle θ. First, we make the following definitions.

Since the transformation to look in Step (3.4) involves only rotation, we assume that T=0. Then, given a rotation of angle θ along an axis J, we define the contribution of the i-th amino acid pair as

$$\text{TM-score}_i(J,\theta) = \frac{1}{1+\left(\frac{\|Ra_i - b_i\|}{d_0}\right)^2}$$

where R is the rotation defined by J and θ. We are interested in how $\text{TM-score}_i(J,\theta)$ changes with respect to θ. Without loss of generality assume that the rotation axis is the y-axis. Suppose b_i has y coordinate h_i and is of minimum distance r_i from be y-axis. Suppose a_i has coordinate (x_i, y_i, z_i) in the new coordinate system. Then, the distance from b_i and a_i after a rotation angle θ is

$$d_i = \sqrt{(x_i - r_i\cos\theta)^2 + (y_i - h_i)^2 + (z_i - r_i\sin\theta)^2}$$

Hence

$$\begin{aligned} d_i^2 &= x_i^2 + y_i^2 + z_i^2 + r_i^2 + h_i^2 - 2y_i h_i 2r_i(x_i \cos\theta - z_i \sin\theta) \\ &= a_i \cdot a_i + b_i \cdot b_i - 2y_i h_i - 2r_i(x_i \cos\theta + z_i \sin\theta) \end{aligned}$$

To simplify the notations, we let

$$c_{i,0} = x_i^2 + y_i^2 + z_i^2 + r_i^2 + h_i^2 - 2y_i h_i$$

$$c_{i,1} = 2r_i x_i$$

$$c_{i,2} = 2r_i z_i$$

Then, $\text{TM-Score}_i(J, \theta)$ can be written as

$$\text{TM-Score}_i(J, \theta) = \frac{d_0^2}{d_0^2 + c_{i,0} - c_{i,1} \cos\theta - c_{i,2} \sin\theta}$$

It is clear that $\text{TM-score}_i(J, \theta)$ is maximized when $-c_1 \cos\theta - c_2 \sin\theta$ is minimized. This happens when

$$x_i \sin\theta = z_i \cos\theta$$

$$\theta = \tan^{-1} z_i / x_i \qquad (6)$$

We denote the set of angles fulfilling Eqn. (6) by $\Omega_i(J)$, and denote this maximum contribution by $\text{Max-TM-score}_i(J)$.

Now consider the possible values of $\text{TM-score}_i(J, \theta)$ within a rotation interval $[\alpha, \alpha + \omega]$, denoted $\text{TM-score}_i(J, [\alpha, \alpha + \omega])$.

In the case that $\Omega_i(J) \cap [\alpha, \alpha + \omega] = \emptyset$,

$$\max\big(\text{TM-score}_i(J,[\alpha,\alpha+\omega])\big) \le \max\{\text{TM-score}_i(J,\alpha),\text{TM-score}_i(J,\alpha+\omega)\}.$$

In the case that $\Omega_i(J) \cap [\alpha,\alpha+\omega] \neq \emptyset$,

$$\max\big(\text{TM-score}_i(J,[\alpha,\alpha+\omega])\big) \le \text{Max-TM-score}_i(J)$$

These two conditions give us a way to obtain an upper-bound to the maximum TM-score obtainable by rotating along the axis J. That is, $\sum_{i=1}^{n}$ Max-TM-score$_i(J)$, or more precisely, $\frac{1}{n}\max_{\theta}\sum_{i=1}^{n}\frac{1}{1+\left(\frac{\|Ra_i-b_i\|}{d_0}\right)^2}$.

To compute this upper-bound, we perform an exhaustive search in a divide and conquer fashion. At the topmost layer we computer $\sum_{i=1}^{n}$ Max-TM-score$_i(J)$ at h intervals, i.e. at the angles $0,\frac{2\pi}{h},\frac{4\pi}{h},\dots,2\pi$. At the subsequent layers, we take two consecutive angles θ from the topmost layer, θ and $\theta+\frac{2\pi}{h}$ say, and further compute TM-score at h intervals within it, i.e. at the angle $\theta,\theta+\frac{2\pi}{h^2},\dots,\theta+\frac{2\pi}{h}$. This is done recursively. The computation for an interval at a specific layer is halted if the interval between two consecutive angles results in a step size smaller than that required, or further refinement will not yield a TM-score larger than the largest computed so far, based on the upper-bound as discussed. The pseudo-codes in Table 3.2 and 3.3 show this computation.

Input: Protein structures A, B, and a rotation axis J.

(1) Set global variable MAX to 0.

(2) Call subroutine Find-Max-TM-score$(J, [0, 2\pi])$.

(3) Output MAX.

Table 3.2: Main routine for computing $\sum$ Max-TM-score$_i(J)$

Input: Protein structures A, B, rotation axis J, and interval $[\theta, \theta + \alpha]$

(1) For each interval X in $[\theta, \theta + \frac{\alpha}{h}], \ldots, [\theta + \frac{(h-1)\alpha}{h}]$, do

(1.1) If the interval X would result in a smaller step size than required, let $\text{MAX} = \max\begin{Bmatrix} \sum_{i=1}^{n} \text{TM-score}_i(J, \theta) \\ \sum_{i=1}^{n} \text{TM-score}_i(J, \theta + \alpha) \\ \text{MAX} \end{Bmatrix}$ and return.

(1.2) Else let $\text{MAX}' = \text{MAX}$ and call Find-Max-TM-score$_i(J, X)$.

(1.3) If $\text{MAX}' > \text{MAX}$, let $\text{MAX} = \text{MAX}'$. (Restore value.)

Table 3.3: Find-Max-TM-Score (J, $[\theta, \theta + \alpha]$)

With the computation of Step (3.4) in Table 3.1 explained, our algorithm is complete. We next compare it to the currently available algorithm, which is listed in Table 2.3, to examine their differences.

3.4 Contrast with the current method

Table 3.4 shows a side-by-side comparison of our new method with the currently available method.

Current method	**Our method**
1. Start with a short (contiguous) fragment. 2. Superpose the fragment to the corresponding residues of the other structure through Kabsch algorithm. 3. Collect all of the residues that are matched closely to form a new fragment. 4. Repeat Steps 2–3 until the new fragment is the same as the old.	1. Start with an arbitrary *R*. 2. Translate structure with semi-optimal *T* obtained through our analysis under *R*. 3. Compute a semi-optimal *R* by through the Kabsch algorithm but without the translation step. 4. Repeat Steps 2–3 until no significant further improvements in TM-score can be obtained.

Table 3.4: Side-by-side comparison of our method with the current method

Several differences can be noted of the new algorithm:

1. At each iteration, it considers the protein structures in their entirety instead of only a set of matching residues.
2. Optimal TM-score is achieved at each iteration, compared to the current method which optimizes the RMSD instead.

In the next chapter, we will compare the new method to the current one in terms of their performances in computing actual TM-score.

CHAPTER 4

RESULTS

4.1 Software preparation

The new algorithm is implemented in C++. The program accepts two input files: each file is to contain a list of coordinates written in the PDB file format – the format used by the Protein Data Bank (Berman *et al.*, 2000).

An example is given in Table 4.1.

```
decoy 1 energy=-1770.2
ATOM      1  N   PHE     1      -6.907  11.411  -5.439
ATOM      2  CA  PHE     1      -6.529  12.598  -5.984
ATOM      3  C   PHE     1      -6.455  12.732  -7.492
ATOM      4  O   PHE     1      -6.716  11.831  -8.203
ATOM      5  CB  PHE     1      -7.517  12.355  -7.157
ATOM      6  CG  PHE     1      -8.853  12.986  -6.850
ATOM      7  CD1 PHE     1      -9.839  12.245  -6.193
ATOM      8  CD2 PHE     1      -9.142  14.273  -7.287
ATOM      9  CE1 PHE     1     -11.097  12.797  -5.949
ATOM     10  CE2 PHE     1     -10.412  14.848  -7.057
ATOM     11  CZ  PHE     1     -11.376  14.082  -6.384
ATOM     12  N   ILE     2      -6.077  13.919  -8.037
ATOM     13  CA  ILE     2      -5.987  14.112  -9.439
ATOM     14  C   ILE     2      -4.547  13.861  -9.376
ATOM     15  O   ILE     2      -3.883  13.901 -10.406
ATOM     16  CB  ILE     2      -6.984  15.015 -10.245
ATOM     17  CG1 ILE     2      -7.086  14.544 -11.710
ATOM     18  CG2 ILE     2      -6.560  16.492 -10.130
ATOM     19  CD1 ILE     2      -8.174  15.271 -12.521
END
```

Table 4.1: Example PDB file content

Each line in the file which starts with the word “ATOM” gives away the 3D coordinate of an amino acid in the protein structure chain. For instance, the first line in Table,

```
ATOM      1  N   PHE     1      -6.907  11.411  -5.439
```

states the coordinate of the first amino acid as (-6.907, 11.411, -5.439). Similarly, the second line gives the coordinate of the second amino acid as (-6.529, 12.598, -5.984). These coordinates have no absolute meaning. They should be understood as only conveying the relative position of each amino acid from the other amino acids.

The program is invoked through the command:

```
> tm2 file1.pdb file2.pdb
```

Here, file1.pdb and file2.pdb are the names of the input PDB files.

The program then computes the TM-score according to the new algorithm proposed in this thesis, and output that score on the console.

If invoked with a “–v” switch,

```
> tm2 –v file1.pdb file2.pdb
```

the program will, in addition, output the superposition for the TM-score.

This superposition is given as a transformation which is to be applied to the set of coordinates in both input files. Table 4.2 shows an example of the transformations given by the program:

```
Transformation for /decoys/1abv_/d2.pdb:
   -Translation (before rotation)
    (-0.50859049, -0.54650653, -0.68786361)
   -Rotation
    [   0.99994680   0.00653589  -0.00797949 ]
    [  -0.00658550   0.99995905  -0.00620671 ]
    [   0.00793860   0.00625893   0.99994890 ]

 Transformation for /decoys/1abv_/d1.pdb:
   -Translation
    (-0.38875880, -0.60755767, -0.78644808)
```

Table 4.2: Transformations to superpose the two structures to obtain the TM-score

Having information of the superposition allowed us to verify the correctness of the program.

The C++ source codes for the program can be obtained from the GitHub repository at: https://github.com/kalngyk/tm2. The GNU C++ compiler is used to compile the source codes into an executable.

The new tool is compared to the current standard tool used to compute TM-scores. The tool is written in Fortran by Zhang and Skolnick and is obtained from https://zhanglab.ccmb.med.umich.edu/TM-score/. The GNU Fortran compiler is used to compile the source codes into an executable.

4.2 Sample preparation

For samples, we use the database of created by the protein structure prediction system called I-TASSER (Wu *et al.*, 2007). The database consists of the structures that I-TASSER predicted out of 56 different known protein structures (called *natives*). For each native, I-TASSER predicted between 6119 to

32000 structures (called *models*). The exact number of models predicted on each native structure, along with the length of the structures, are shown in Table 4.3.

Native	#Models	length	Native	#Models	Length
1abv_	12500	103	1mkyA3	6119	81
1af7_ _	12499	72	1mla_2	12500	70
1ah9_	27498	63	1mn8A	12500	84
1aoy_	32000	65	1n0uA4	12499	69
1b4bA	12500	71	1ne3A	12500	56
1b72A	12499	49	1no5A	12500	93
1bm8_	20000	99	1npsA	20000	88
1bq9A	20000	53	1o2fB_	12500	77
1cewI	19830	108	1of9A	20000	77
1cqkA	19999	101	1ogwA_	19998	72
1csp_	12500	67	1orgA	20000	118
1cy5A	32000	92	1pgx_	20000	59
1dcjA_	20000	73	1r69_	20000	61
1di2A_	20000	69	1sfp_	19985	111
1dtjA_	20000	74	1shfA	20000	59
1egxA	20000	115	1sro_	20000	71
1fadA	12599	92	1ten_	20000	87
1fo5A	20000	85	1tfi_	32000	47
1g1cA	19997	98	1thx_	32000	108
1gjxA	12500	77	1tif_	12500	59
1gnuA	17533	117	1tig_	12500	88
1gpt_	32000	47	1vcc_	20000	76
1gyvA	11508	117	256bA	20000	106
1hbkA	20000	89	2a0b_	32000	118
1itpA	12500	68	2cr7A	12500	60
1jnuA	20000	104	2f3nA	19999	65
1kjs_	20000	74	2pcy_	20000	99
1kviA	20000	68	2reb_2	12500	60

Table 4.3: Samples from I-TASSER

The database was originally downloaded on the 24th July of 2009 from I-TASSER's website, http://zhang.bioinformatics.ku.edu/I-TASSER/decoys/. The website is no longer available. Furthermore, a minor part of the data in our

collection has been damaged due to file corruption. Table 4.4 shows the current state of the collection. The current experiments are performed on this set of data.

Native	#Models	Current #	Native	#Models	Current #
1abv_	12500	12500	1mkyA3	6119	6119
1af7_ _	12499	12472	1mla_2	12500	12500
1ah9_	27498	27498	1mn8A	12500	12475
1aoy_	32000	32000	1n0uA4	12499	12499
1b4bA	12500	12500	1ne3A	12500	12500
1b72A	12499	12463	1no5A	12500	12500
1bm8_	20000	20000	1npsA	20000	20000
1bq9A	20000	20000	1o2fB_	12500	12500
1cewI	19830	19830	1of9A	20000	20000
1cqkA	19999	19999	1ogwA_	19998	19998
1csp_	12500	12500	1orgA	20000	19982
1cy5A	32000	32000	1pgx_	20000	20000
1dcjA_	20000	20000	1r69_	20000	20000
1di2A_	20000	20000	1sfp_	19985	19985
1dtjA_	20000	20000	1shfA	20000	20000
1egxA	20000	19279	1sro_	20000	20000
1fadA	12599	12500	1ten_	20000	19976
1fo5A	20000	19975	1tfi_	32000	32000
1g1cA	19997	19997	1thx_	32000	32000
1gjxA	12500	12500	1tif_	12500	12500
1gnuA	17533	17517	1tig_	12500	12500
1gpt_	32000	31954	1vcc_	20000	20000
1gyvA	11508	11508	256bA	20000	16910
1hbkA	20000	20000	2a0b_	32000	22930
1itpA	12500	179	2cr7A	12500	12500
1jnuA	20000	65	2f3nA	19999	19999
1kjs_	20000	19971	2pcy_	20000	20000
1kviA	20000	19969	2reb_2	12500	12500

Table 4.4: Current state of samples from I-TASSER (cases that suffered very significant file loss are colored red)

4.3 TM-score comparison

For each native structure, the TM-score between it and each of its predicted models is computed, and then an average of these TM-scores is obtained. This results in 56 average TM-score, each for one native structure. These averages, obtained using the current TM-score tool and our new TM-score tool respectively, are shown in Table 4.5.

Native	Current	New	Native	Current	New
1abv_	0.29089	0.29201	**1mkyA3**	**0.39476**	**0.39807**
1af7_ _	0.37797	0.37940	1mla_2	0.59582	0.59741
1ah9_	0.51037	0.51175	1mn8A	0.28868	0.29002
1aoy_	0.66534	0.66603	**1n0uA4**	**0.47075**	**0.47287**
1b4bA	0.43531	0.43701	**1ne3A**	**0.36134**	**0.36413**
1b72A	0.56623	0.56810	1no5A	0.41955	0.42045
1bm8_	0.30003	0.30117	1npsA	0.72768	0.72811
1bq9A	**0.36093**	**0.36309**	1o2fB_	0.35697	0.35850
1cewI	0.18051	0.18156	1of9A	0.53111	0.53280
1cqkA	0.81051	0.81081	1ogwA_	0.67277	0.66040
1csp_	0.69542	0.69625	1orgA	0.74720	0.74756
1cy5A	0.85736	0.85754	**1pgx_**	**0.47822**	**0.48039**
1dcjA_	0.35526	0.35622	1r69_	0.71156	0.71243
1di2A_	0.73412	0.71599	1sfp_	0.72730	0.72754
1dtjA_	0.75502	0.75568	1shfA	0.58400	0.58552
1egxA	0.74859	0.74904	1sro_	0.62662	0.62753
1fadA	0.57726	0.57820	1ten_	0.78512	0.78543
1fo5A	**0.52384**	**0.52571**	1tfi_	0.47232	0.47429
1g1cA	0.75538	0.75570	1thx_	0.78107	0.78135
1gjxA	0.34868	0.35011	1tif_	0.31027	0.31152
1gnuA	0.53802	0.53861	1tig_	0.47187	0.47327
1gpt_	**0.49780**	**0.50002**	1vcc_	0.35366	0.35534
1gyvA	0.74122	0.74149	256bA	0.75671	0.75716
1hbkA	0.60986	0.61082	2a0b_	0.76548	0.76586
1itpA	0.30651	0.30764	2cr7A	0.42824	0.42999
1jnuA	0.69819	0.69869	2f3nA	0.69283	0.69397
1kjs_	0.38041	0.38190	2pcy_	0.62193	0.62244
1kviA	0.69722	0.69788	2reb_2	0.37690	0.37850

Table 4.5: Average TM-score obtaining using both the current method and the new method proposed (each comparison is colored green if the new method has an improvement above 0.001, and set in bold font if the new method has an improvement above 0.002)

Except in the cases of the two native structures 1di2A_ and 1ogwA_, the new method obtained TM-scores that improved on those from the current method.

Larger improvements can be observed if instead of considering averages, we look at an individual model. For some models, the difference in the TM-score obtained using the new method can be larger than 0.01 from the value obtained using the current method. The largest 20 such differences are listed in Table 4.6, together with the corresponding filenames of the models.

Native	Model file name	TM-score		Improvement
		Current	New	
1cewI	d18838.pdb	0.1544	0.1845	0.0301
1ne3A	d7337.pdb	0.2492	0.2682	0.0190
1mkyA3	d12097.pdb	0.3568	0.3743	0.0175
1ne3A	d7338.pdb	0.2627	0.2801	0.0174
1ne3A	d9836.pdb	0.2528	0.2701	0.0173
1tfi_	d31506.pdb	0.2597	0.2765	0.0168
1ne3A	d4902.pdb	0.2485	0.2650	0.0165
1tfi_	d23509.pdb	0.2646	0.2810	0.0164
1mkyA3	d12100.pdb	0.3610	0.3773	0.0163
1cewI	d13371.pdb	0.1720	0.1883	0.0163
1ne3A	d7229.pdb	0.2349	0.2511	0.0162
1ne3A	d12388.pdb	0.2505	0.2664	0.0159
1ne3A	d8128.pdb	0.2383	0.2540	0.0157
1ne3A	d7283.pdb	0.2441	0.2598	0.0157
1ne3A	d9347.pdb	0.2462	0.2617	0.0155
1mkyA3	d4534.pdb	0.3734	0.3887	0.0153
1ne3A	d11198.pdb	0.2233	0.2386	0.0153
1ne3A	d5629.pdb	0.2420	0.2572	0.0152
1ne3A	d11664.pdb	0.2399	0.2546	0.0147
1mkyA3	d10192.pdb	0.3748	0.3893	0.0145

Table 4.6: Models where the TM-scores computed using the new method improved significantly over those obtained using the current method (only the top 20 cases are shown)

4.4 Runtime comparison

To examine if the new tool runs in a reasonable time, tests were performed on a PC with the following specification:

- Intel Pentium G4400 3.3 GHz CPU
- 4GB RAM
- Cygwin environment (Windows 7 host)

Native	Current	New	Ratio	Native	Current	New	Ratio
1abv_	5	56	11.2	1mkyA3	6	41	6.8
1af7_ _	11	46	4.2	1mla_2	9	47	5.2
1ah9_	5	28	5.6	1mn8A	10	19	1.9
1aoy_	5	22	4.4	1n0uA4	7	50	7.1
1b4bA	3	33	11.0	1ne3A	10	58	5.8
1b72A	11	20	1.8	1no5A	10	21	2.1
1bm8_	7	75	10.7	1npsA	35	36	1.0
1bq9A	8	35	4.4	1o2fB_	11	33	3.0
1cewI	6	69	11.5	1of9A	10	16	1.6
1cqkA	7	20	2.9	1ogwA_	12	24	2.0
1csp_	4	15	3.8	1orgA	8	30	3.8
1cy5A	9	17	1.9	1pgx_	10	26	2.6
1dcjA_	6	37	6.2	1r69_	32	32	1.0
1di2A_	5	15	3.0	1sfp_	12	23	1.9
1dtjA_	6	16	2.7	1shfA	16	25	1.6
1egxA	19	34	1.8	1sro_	7	32	4.6
1fadA	7	52	7.4	1ten_	8	53	6.6
1fo5A	23	45	2.0	1tfi_	8	53	6.6
1g1cA	7	21	3.0	1thx_	28	31	1.1
1gjxA	4	46	11.5	1tif_	39	45	1.2
1gnuA	21	47	2.2	1tig_	6	34	5.7
1gpt_	32	36	1.1	1vcc_	8	20	2.5
1gyvA	7	20	2.9	256bA	10	25	2.5
1hbkA	8	31	3.9	2a0b_	5	36	7.2
1itpA	9	36	4.0	2cr7A	6	41	6.8
1jnuA	34	37	1.1	2f3nA	9	47	5.2
1kjs_	32	50	1.6	2pcy_	10	19	1.9
1kviA	32	28	0.9	2reb_2	7	50	7.1

Table 4.7: Runtime of the new tool compared to the current tool (seconds used per 100 TM-score computations)

To perform this test, 100 models were randomly selected for each native structure (except for 1jnuA where there are less than 100 models), and the total time is taken for the tool to compute the 100 TM-scores between these models and the native structure were recorded. These total times are shown in Table 4.6.

On average, the new method took 4.2 times the time required by the current method. However, in several cases, namely, 1abv_, 1b4bA, 1bm8_, 1cewI, 1gjxA, the new method took more than 10 times the time taken by the current method. On the other hand, the time is taken by the tool for each computation never exceeded a second, which is reasonable for most routine usage.

4.5 Discussions

The tests in this section demonstrated that the new tool provides better TM-score computation than the currently available tool. The runtime required of the new tool was also shown to be within reasonable limits for routine use.

It can hence be concluded that the proposed algorithm has achieved its original aim, that is, to compute better TM-score at a reasonable time for either replacing the current tool or be used to verify the correctness of the TM-scores found by the current tool.

CHAPTER 5

CONCLUSION

This research proposed a new algorithm for the computation of the TM-score, a popular scoring function for measuring the similarity between two protein structures among the protein structure prediction community. The algorithm uses a different strategy for iteration from the currently available algorithm – whereas at each iteration, the current algorithm finds a superposition of only a *subset of the structures* which optimizes the *RMSD*, the new method aims to find a superposition of the *entire structures* which optimizes *TM-score*.

A C++ tool was implemented based on the algorithm and made available to researchers.

Test results based on publicly available database showed the tool to give better TM-scores than the currently available tool in every case except for a few. The runtime required by the tool is within a fraction of a second and can be used routinely, as a replacement for the current tool, or as a verifier in situations where accuracy is important.

For future work, it is worth investigating whether the iteration framework espoused by the new algorithm can be extended to the computation of the other scores such as GDT and MaxSub.

In the future, we will investigate the possibilities to define a new measurement for protein structures. The RMSD measurement is metric. However, the range is not between 0 and 1. On the other hand, GDT, MaxSub, and TM-

score are between 0 and 1, and they are not metric. We find new definitions of distances for protein structures, which are both metric and with the range from 0 to 1.

REFERENCES

Alberts, B., Johnson, A., Lewis, J., Morgan, D., Raff, M., Roberts, K. And Walter, P., 2014. Molecular Biology of the Cell, 6th ed. New York: Garland Science.

Kufareva, I. And Abagyan, R., 2012. Methods of protein structure comparison, Methods in Molecular Biology, 857, pp. 231–257.

Kabsch, W., 1976. A solution for the best rotation to relate two sets of vectors, Acta Crystallographica, A32, pp. 922–923.

Zemla, A., Venclovas, C., Moult, J., Fidelis, K., 1999. Processing and analysis of CASP3 protein structure predictions. Proteins. S3, pp. 22–29.

Zemla, A., 2003. LGA – a method for finding 3D similarities in protein structures, Nucleic Acids Research, 31(13), pp. 3370–3374.

Siew, N., Elofsson, A., Rychlewski, L. And Fischer, D., 2000. MaxSub: an automated measure for the assessment of protein structure prediction quality, Bioinformatics, 16(9), pp. 776–85.

Zhang, Y. And Skolnick, J., 2004. Scoring function for automated assessment of protein structure template quality, Proteins, 57, pp. 702–710.

Sanger, F. And Coulson, A. R., 1975. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. Journal of Molecular Biology, 94 (3), pp. 441–448.

Ronaghi, M., Uhlén, M. And Nyrén, P., 1998. A sequencing method based on real-time pyrophosphate. Science. 281 (5375), pp. 363–365.

Needleman, S. B. And Wunsch, C. D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of Molecular Biology, 48 (3), pp. 443–453.

Smith, T. F., and Waterman, M. S., 1981. Identification of Common Molecular Subsequences, Journal of Molecular Biology, 147, pp. 195–197.

Dorn, M., e Silva, M. B., Buriol, L. S. And Lamb, L. C., 2014. Three-dimensional protein structure prediction. Computational Biology and Chemistry. 53, pp. 251–276.

Nishikawa, K., Ooi, T., Isogai, Y. And Saitô, N., 1972. Tertiary Structure of Proteins. I. Representation and Computation of the Conformations, Journal of the Physical Society of Japan. 32, pp. 1331–1337.

Rao, S.T. And Michael Rossmann, G., 1973. Comparison of super-secondary structures in proteins, Journal of Molecular Biology, 76(2), pp. 241–250.

Li, S. C., Bu, D., Xu, J. And Li, M., 2008, Finding largest well-predicted subset of protein structure models, in Proceedings of the 19th annual symposium on Combinatorial Pattern Matching, CPM'08, Springer-Verlag, pp. 44–55.

Ortiz, A. R., Strauss, C. E. And Olmea, O., 2002. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison, Protein Science, 11(11), pp. 2606–2621.

Kihara, D. And Skolnick, J., 2003. The PDB is a covering set of small protein structures, Journal of Molecular Biology, 334(4), pp. 793-802.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E., 2000. The Protein Data Bank, Nucleic Acids Research, 28(1), pp. 235–42.

Wu, S., Skolnick, J., Zhang, Y., 2007. Ab initio modeling of small proteins by iterative TASSER simulations, *BMC Biology*, 5(17).