**TEMPLATE-BASED METHODS FOR**

**PROTEIN MODEL QUALITY ASSESSMENT**

A Thesis

Presented to

The Faculty of the Graduate School

University of Missouri-Columbia

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Wenbo Wang

Dr. Yi Shang, Thesis Supervisor

December 2016

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

TEMPLATE-BASED METHODS FOR PROTEIN MODEL QUALITY ASSESSMENT

Presented by Wenbo Wang,

A candidate for the degree of Master of Computer Science,

And hereby certify that, in their opinion, it is worthy of acceptance.

**Dr. Yi Shang**

**Dr. Dong Xu**

**Dr. Ioan** **Kosztin**

**ACKNOWLEDGEMENT**

First of all, I would like to thank my advisor, Dr. Yi Shang for all of his guidance, support and patience for all these time. I remember he was the first professor I have ever met when I come to University of Missouri. He has been helping me getting through all kinds of obstacles ever since then. It has been a great honor and privilege to be one of his students and this thesis would not be possible without him. I also would like to thank Dr. Dong Xu for the professional opinions and valuable advices he gave me. His kind help allowed me to avoid many detours and mistakes. And, I would like to thank Dr. Ioan Kosztin not only for being on the thesis committee but also for lending his expertise and bringing me up to speed to a completely new field.

I would like to specially express my gratitude to my parents who have been extremely caring and supportive. They have always been there for me no matter what has happened. There is no words in the world would be enough to express how much I want to thank them.

Finally, I would like to thank everyone in the protein group for all their help and support. It has been a great pleasure working with them.

**TABLE OF CONTENTS**

[ACKNOWLEDGEMENT II](#_Toc468990875)

[TABLE OF CONTENTS III](#_Toc468990876)

[LIST OF FIGURES VI](#_Toc468990877)

[LIST OF TABLES VIII](#_Toc468990878)

[ABSTRACT 1](#_Toc468990879)

[1. INTRODUCTION 3](#_Toc468990880)

[2. RELATED WORK 5](#_Toc468990881)

[2.1 The basic ideas behind current methods 5](#_Toc468990882)

[2.2 BLOSUM table 5](#_Toc468990883)

[2.3 GDT-TS 6](#_Toc468990884)

[2.4 Early attempts template-based QA method 6](#_Toc468990885)

[3. CORE ALGORITHM 7](#_Toc468990886)

[3.1 MUfoldQA\_S 7](#_Toc468990887)

[3.2 MUfoldQA\_C 9](#_Toc468990888)

[4. IMPLEMENTATION 11](#_Toc468990889)

[4.1 Web interface 11](#_Toc468990890)

[4.2 Alignment generator 14](#_Toc468990891)

[4.3 Core-algorithm helper 16](#_Toc468990892)

[4.4 MUfoldQA core program 17](#_Toc468990893)

[4.4.1 Main module 17](#_Toc468990894)

[4.4.2 Template JSON loader 18](#_Toc468990895)

[4.4.3 Stage 1 alignment generator 19](#_Toc468990896)

[4.4.4 Stage 1 score calculator 22](#_Toc468990897)

[4.4.5 Stage 2 reference model generator 23](#_Toc468990898)

[4.4.6 Stage 2 score calculator 23](#_Toc468990899)

[4.4.7 advanced decoy loader 24](#_Toc468990900)

[4.5 GDT-TS calculator 24](#_Toc468990901)

[5. EXPERIMENTS AND RESULTS 26](#_Toc468990902)

[5.1 Data Set 26](#_Toc468990903)

[5.1.1 Targets 26](#_Toc468990904)

[5.1.2 Server models 27](#_Toc468990905)

[5.2 Experiment procedure 28](#_Toc468990906)

[5.2.1 Speed-up test 28](#_Toc468990907)

[5.2.2 Accuracy test 28](#_Toc468990908)

[5.3 Evaluation Parameters 28](#_Toc468990909)

[5.4 Collecting data for other algorithms 29](#_Toc468990910)

[5.5 Results and analysis 29](#_Toc468990911)

[5.5.1 Speedup test 29](#_Toc468990912)

[5.5.2 Overall ranking 31](#_Toc468990913)

[5.5.2.1 Stage 1 ranking 31](#_Toc468990914)

[5.5.2.2 Stage 2 ranking 34](#_Toc468990915)

[5.5.3 Comparison between different methods 36](#_Toc468990916)

[5.5.3.1 Comparison between MUfoldQA\_S and MUfoldQA\_C 36](#_Toc468990917)

[5.5.3.2 Comparison between MUfoldQA\_S and other quasi-single model QA methods ………………………………………………………………………39](#_Toc468990918)

[5.5.3.2.1 Comparison between MUfoldQA\_S and MQAPsingle\_A 40](#_Toc468990919)

[5.5.3.2.2 Comparison between MUfoldQA\_S and ModFOLD5\_single 41](#_Toc468990920)

[5.5.3.2.3 Comparison between MUfoldQA\_S and myprotein-me 44](#_Toc468990921)

[5.5.3.3 Comparison between MUfoldQA\_C and other multi-model QA methods ………………………………………………………………………46](#_Toc468990922)

[5.5.3.3.1 Comparison between MUfoldQA\_C and Pcons-net 46](#_Toc468990923)

[5.5.3.3.2 Comparison between MUfoldQA\_C and MULTICOM-REFINE ……………………………………………………….………48](#_Toc468990924)

[5.5.3.3.3 Comparison between MUfoldQA\_C and DAVIS-QAconsensus ……………………………………………………………….50](#_Toc468990925)

[5.5.4 Robustness comparison 53](#_Toc468990926)

[6. CONCLUSION AND FUTURE WORK 55](#_Toc468990927)

[REFERENCES 56](#_Toc468990928)

**LIST OF FIGURES**

[Figure 1 Web Interface 11](#_Toc468984739)

[Figure 2 Sample web request 12](#_Toc468984740)

[Figure 3 Sample Request email 13](#_Toc468984741)

[Figure 4 Sample result email 14](#_Toc468984742)

[Figure 5 Raw blast results 15](#_Toc468984743)

[Figure 6 Parsed Blast result 15](#_Toc468984744)

[Figure 7 Sample report message 16](#_Toc468984745)

[Figure 10 Sample HTTP traffic 26](#_Toc468984746)

[Figure 11 Speed comparison 30](#_Toc468984747)

[Figure 14 Stage 1 Per target correlation comparison between MUfoldQA\_C and MUfoldQA\_S 37](#_Toc468984748)

[Figure 15 Stage 2 Per target correlation comparison between MUfoldQA\_C and MUfoldQA\_S 39](#_Toc468984749)

[Figure 16 Stage 1 Per target correlation comparison between MUfoldQA\_S and MQAPsingle\_A 40](#_Toc468984750)

[Figure 17 Stage 2 Per target correlation comparison between MUfoldQA\_S and MQAPsingle\_A 41](#_Toc468984751)

[Figure 18 Stage 1 Per target correlation comparison between MUfoldQA\_S and ModFOLD5\_single 42](#_Toc468984752)

[Figure 19 Stage 2 Per target correlation comparison between MUfoldQA\_S and ModFOLD5\_single 43](#_Toc468984753)

[Figure 20 Stage 1 Per target correlation comparison between MUfoldQA\_S and myprotein-me 44](#_Toc468984754)

[Figure 21 Stage 2 Per target correlation comparison between MUfoldQA\_S and myprotein-me 45](#_Toc468984755)

[Figure 22 Stage 1 Per target correlation comparison between MUfoldQA\_C and Pcons-net 46](#_Toc468984756)

[Figure 23 Stage 2 Per target correlation comparison between MUfoldQA\_C and Pcons-net 47](#_Toc468984757)

[Figure 24 Stage 1 Per target correlation comparison between MUfoldQA\_C and MULTICOM-REFINE 48](#_Toc468984758)

[Figure 25 Stage 2 Per target correlation comparison between MUfoldQA\_C and MULTICOM-REFINE 49](#_Toc468984759)

[Figure 26 Stage 1 Per target correlation comparison between MUfoldQA\_C and DAVIS-QAconsensus 50](#_Toc468984760)

[Figure 27 Stage 2 Per target correlation comparison between MUfoldQA\_C and DAVIS-QAconsensus 52](#_Toc468984761)

[Figure 28 Ranking of stage 1 and stage 2 sorted by average of both 53](#_Toc468984762)

## LIST OF TABLES

[Table 1 Template database data structure 19](#_Toc468984763)

[Table 2 Weight Lookup Table 22](#_Toc468984764)

[Table 3 Ranking of CASP 11 stage 1 dataset 33](#_Toc468984765)

[Table 4 Ranking of CASP 11 stage 2 dataset 36](#_Toc468984766)

**ABSTRACT**

Protein structure prediction is an important open problem in the bioinformatics filed. One of the difficulties of solving this problem is to develop an effective approach to evaluate the quality of the models been generated. Various of quality assessment (QA) methods have been developed and tested in CASP (Critical Assessment of Techniques for Protein Structure Prediction) competition. But most of them are either not accurate enough to be useful or not robust enough for different kinds of model sets.

In pursuit of a balance between high accuracy and robustness, two QA methods have been developed: MUfoldQA\_S and MUfoldQA\_C. MUfoldQA\_S is a quasi-single model QA method. It assesses the quality of a predicted model based on the structures of proteins with similar sequence. These similar proteins, called templates, are found from the PDB database by using sequence search. We calculate the pairwise GDT-TS between the input model and the templates. Then, for each c-alpha position of the model, a score is calculated as the weighted average of the template GDT-TS values, weighted by a BLOSUM-based heuristic. Finally, the model score is the average of all c-alpha position scores. The other method, MUfoldQA\_C, is a 2-stage multi-model QA method combining the idea behind MUfoldQA\_S and consensus. Stage 1 evaluates the quality of each c-alpha position of the reference models based on their similarity to the templates. Stage 2 evaluates the quality of the given predicted model based on its similarity to the reference models and the quality of the reference models. Both methods have been tested on CASP 11 dataset. MUfoldQA\_S performs significantly better than ProQ2 and MUfoldQA\_C also outperforms the naïve consensus method.

1. **INTRODUCTION**

Proteins are macromolecules performing varieties of functions within organisms [1]. For a given protein, its biological functionality is highly relevant to its conformation [2]. Therefore, knowing the 3D structure of the protein is important for the analysis of its functionality [3]. Current experimental methods to acquire protein structure include electron microscopy, protein crystallography and nuclear magnetic resonance [4]. The problem is, The experimental approach of determine protein structure is both expensive and time consuming [5]. As of December 8, 2016, UniProtKB/TrEMBL database (<http://www.ebi.ac.uk/uniprot/TrEMBLstats>) contains 71,002,161 protein sequences entries, but only 124,928 of which have their structure determined by experiment (<http://www.rcsb.og>). Computational protein structure prediction is the only way to bridge this gap between the known sequence and known structure [6]. With the steady progress during the last few decades [7], the ever increasing number of native-like predicted models we are capable of generating is making model selection comparably difficult [8].

The problem of evaluating the quality of the predicted model, also known as quality assessment, has received lots of attention from researchers [9]. Numbers of methods has been developed and tested [10]. The proposed methods can be divided to three major categories: single model QA, quasi-single model QA, and multi-model QA. The single model QA methods only uses one decoy when providing the score while multi-model QA methods could use multiple decoys from the pool. The quasi-single model QA methods are in between, it only uses one decoy from the pool, but might also use its own predicted model.

In this thesis, we proposed 2 QA methods, MUfoldQA\_S and MUfoldQA\_C. The former is a quasi-single model QA method that directly uses fragments of other proteins of similar sequence to estimate the quality of predicted model. The MUfoldQA\_C is a multi-model QA method that uses the local score from MUfoldQA\_S as weight and top model from the pool as reference model. It calculates the weighted consensus score for each predicted model as output. The chapter 2 describes the algorithms themselves and chapter 3 gives some details about the real-world implementation. And finally in chapter 5 the algorithm was tested against other popular QA methods of the same category to see the performance of the proposed algorithm.

1. **RELATED WORK**
	1. **The basic ideas behind current methods**

As mentioned above, the QA methods could be categorized to single model QA, quasi-single model QA, and multi-model QA. The single model QA methods could only use the predicted model itself, so, the most common approach for these methods are using a combination of physical statistics and machine learning methods. These some famous ones fall into this category including but not limited to dDFIRE [11], DOPE [12], RW [13], RAPDF [14], OPCU-C$α$ [15] as well as ProQ2[16]. Quasi-single model QA methods tends to generate its own model and use these model to score the decoy. For example, the MQAPsingleA first utilizes GeneSilico fold prediction meta-server [17] to generate about one hundred reference models and then score the models by calculating the average GDT-TS distance between the model and reference models These methods may also take advantage of the information from other single model QA methods. For instance, the MQAPsingleB is a combination of eighty per cent MQAPsingleA and twenty per cent MQAPsingleC. In which MQAPsingleC is a single model QA using linear regression [18]. The multi-model QA methods tends to take advantage of the fact that they are allowed to also use other decoys in the pool. For example, Davis-QAconsensus uses the naïve consensus algorithm which is computing the average of GDT-TS value from all other models in the pool [9].

* 1. **BLOSUM table**

In the algorithm, we used a BLOSUM [19] based heuristic value as the weight in MUfoldQA\_S as well as the first stage of MUfoldQA\_C. BLOSUM standards for BLOcks SUbstitution Matrix. It was generated by counting the relative frequencies of amino acids as well as their substitution probabilities and then calculate the logarithm of odds score for each substation pair.

* 1. **GDT-TS**

GDT-TS stands for global distance test total score. This score has the range between [0, 1] with 1 been most identical. It is calculated by first superimposing two protein 3D structures, then count the percentage of corresponding c-alpha atom pairs whose distance falls in the cut-off value of 1,2,4 and 8 angstroms. The final score is the average of these values. It can be represented as below:

$$GDT-TS=\frac{(P\_{d<1}+P\_{d<2}+P\_{d<4}+P\_{d<8})}{4}$$

In which $P\_{d<L} $is Percentage of c-alpha is within L angstrom distance from the correct position after superimpose

GDT-TS value is one of the most important measurement of the difference two protein models.

* 1. **Early attempts template-based QA method**

There are some earlier attempts to directly use templates to score the predicted model. One of the most successful one is TASSER-QA [20]. They used the sliding window of 9 residues to scan and score the predicted model. The problem is that GDT-TS was no sensitive enough under such small protein length, so they have to use RMSD instead, but RMSD score itself is only a shy of 0.5 Pearson correlation to the golden standard GDT-TS, the score we actually want to predict. Also, since RMSD and GDT-TS has very different range and meaning, the mapping of two scores is also another problem. Although both TASSER-QA and MUfoldQA algorithm family are template-based, our algorithm contains very different technology set, including but not limited to adaptive template length, GDT-TS style score calculation, non-linear score combination.

1. **CORE ALGORITHM**
	1. **MUfoldQA\_S**

The input is a target protein sequence and a predicted model, and the output is a quality score of the model in the range of 0 and 1, with 1 being the highest quality - the same as its native structure. The method consists of the following 4 major steps:

1. Search PDB database using Blast [21] and HHsearch [22] to find up to 20 similar proteins, i.e., the templates.
2. Calculate GDT-TS value between the model and each template.
3. For each template, calculate a heuristic weight for each c-alpha position of the template based on the BLOSUM value of the pair of c-alphas at this position in the template and target sequence.
4. Calculate the final model score as the average of all c-alpha position scores, which are weighted GDT-TS values of all available templates for each position.

Here are some details of each step:

 **Step 1.** Use the target protein sequence to search in the PDB database with Blast and HHsearch, respectively, to find similar proteins as templates. Sort them based on scores calculated using the following formula, then choose the top 10 of Blast templates and HHsearch templates, separately:

$$SortScore=(3-log\_{10}E)∙I∙C$$

where E-value (*E*) and the percentage of identical sequences (*I*) are returned by Blast or HHsearch and cover rate (*C*) is the ratio of the length of template sequence to the length of target sequence.

 **Step 2.** Calculate GDT-TS score ($S\_{i}, i=1, 20$) between the input model and each of the 20 templates.

 **Step 3.** For each of the 20 templates, compare its c-alpha sequence with the c-alpha sequence of the target protein. For each pair of c-alphas in the corresponding position of the two sequences, retrieve the BLOSUM value of them (*B*) and use the following formula to calculate a heuristic weight,$ W\_{i,j}$,

$$W\_{i,j}=2^{B}$$

where$ i\in m\_{j}, m\_{j}$ is the set of indices of the templates that have valid value at that c-alpha position, and $j=1 to n, $*n* is the number of c-alphas in the target protein.

 **Step 4.** For each c-alpha position, calculate the weighted average score of GDT-TS of all templates with valid value at that position. Then, the final model score is the simple average of all c-alpha position scores.

$$Score=\frac{1}{n}\sum\_{j=1}^{n}\frac{\sum\_{i\in m\_{j}}^{}W\_{i,j}S\_{i}}{\sum\_{i\in m\_{j}}^{}W\_{i,j}}$$

where *mj* is the set of indices of the templates that have valid value at that c-alpha position and *n* is the number of c-alphas in the target protein.

* 1. **MUfoldQA\_C**

The input is a target protein sequence (*Q*) of length *n* (*n* c-alpha atoms), a predicted model to be scored (*M*), and *r* reference models ($R\_{i}, i=1, r$.Could be other predicted models). The output is a quality score of the model in the range of 0 and 1, with 1 being the highest quality - the same as its native structure. The method consists of the following 5 major steps:

 **Step 1.** Use *Q* to search in the PDB database with Blast and HHsearch, respectively, to find similar proteins as templates. Sort them based on scores calculated using the following formula, then choose the top 10 of Blast templates and HHsearch templates separately to form a set of 20 templates (*T*).

$$SortScore=(3-log\_{10}E)∙I∙C$$

where E-value (*E*) and the percentage of identical sequences (*I*) are returned by Blast or HHsearch and cover rate (*C*) refers to the ratio of the length of template sequence to the length of target sequence.

 **Step 2.** For each reference model $R\_{i}$, call subroutine *Evaluate(Q,* $R\_{i}, $*T)* to evaluate the quality of $R\_{i} $and generate a weight array $H\_{ij}, j=1, n$, consisting of a weight for each c-alpha position in $R\_{i}$.

 **Step 3.** Sort reference models by the average of all elements in its weight array, and choose up to 100 top reference models ($R\_{i}$, *i*=1, *v*. *v*$ \leq 100$)

 **Step 4.** Calculate GDT-TS between *M* and each top reference model $R\_{i}$. Let $G\_{i}, i=1, v$, represent the GDT-TS value vector.

 **Step 5.** For each c-alpha position in *M*, calculate the weighted average of $G\_{i} $using weight $H\_{ij}$. Then, the final model score is the simple average of all c-alpha position scores:

$$Score=\frac{1}{n}\sum\_{j=1}^{n}\frac{\sum\_{i=1}^{v}H\_{ij}G\_{i}}{\sum\_{i=1}^{v}H\_{ij}}$$

**Subroutine** *Evaluate (Q,* $R\_{i}, $*T)*

The subroutine evaluates the quality of a 3-D model $R\_{i} $of a protein sequence *Q* of length *n* based on a set of 20 templates *T*, and generate a weight array $H\_{ij}, j=1, n$, consisting of a weight for each c-alpha position in $R\_{i}$. It has the following 3 major steps:

 **Step 1.** Calculate GDT-TS value ($S\_{k}, k=1, 20$) between the reference model and each of the 20 templates in *T*.

 **Step 2.** For each of the 20 templates, compare its c-alpha sequence with the c-alpha sequence of *Q*. For each pair of c-alpha atoms in the corresponding position of the two sequences, retrieve the BLOSUM3 value of them (*B*) and use the following the formula to calculate a heuristic weight,$ W\_{kj}$,

$$W\_{kj}=2^{B}$$

where$ k\in m\_{j}. $*mj* is the set of indices of the templates that have valid value at that c-alpha position, and $j=1, n$.

 **Step 3.** For each c-alpha position in $R\_{i}$, calculate the weighted average score ($H\_{ij}$) using all templates with valid value at that position.

$$H\_{ij}=\frac{\sum\_{k\in m\_{j}}^{}W\_{kj}S\_{k}}{\sum\_{k\in m\_{j}}^{}W\_{kj}} (j\in \left[1,n\right])$$

where *mj* is the set of indices of the templates that have valid value at that c-alpha position.

1. **IMPLEMENTATION**

Based on above algorithm, a fully automatic server was implemented to perform quality assessment without human-intervention. The whole systems consist of five major modules:

* 1. **Web interface**

Web interface was written in PHP. It provides the accessibility to the program for anyone who is interested. Below is a snapshot of the webpage:



**Figure 1 Web Interface**

It also accepts requests coded in URL:

|  |
| --- |
| http://\*\*\*/mufold\_qa1.php?SEQUENCE=AFCDELMKDTKTW&EMAIL=models@predictioncenter.org&TARGET=TargetName& |

**Figure 2 Sample web request**

After it received a request, it sends a notification to the server administrator:

|  |
| --- |
| Submission for MUfold\_QAA job was submitted at Mon Jul 11 14:24:50 CDT 2016Email:models@predictioncenter.orgTarget Name:T0942TarBall Link:<http://predictioncenter.org/download_area/CASP12/server_predictions/T0942.stage1.3D.srv.tar.gz>Sequence:MFRQLKKNLVATLIAAMTIGQVAPAFADSADTLPDMGTSAGSTLSIGQEMQMGDYYVRQLRGSAPLINDPLLTQYINSLGMRLVSHANSVKTPFHFFLINNDEINAFAFFGGNVVLHSALFRYSDNESQLASVMAHEISHVTQRHLARAMEDQQRSAPLTWVGALGSILLAMASPQAGMAALTGTLAGTRQGMISFTQQNEQEADRIGIQVLQRSGFDPQAMPTFLEKLLDQARYSSRPPEILLTHPLPESRLADARNRANQMRPMVVQSSEDFYLAKARTLGMYNSGRNQLTSDLLDEWAKGNVRQQRAAQYGRALQAMEANKYDEARKTLQPLLAAEPGNAWYLDLATDIDLGQNKANEAINRLKNARDLRTNPVLQLNLANAYLQGGQPQEAANILNRYTFNNKDDSNGWDLLAQAEAALNNRDQELAARAEGYALAGRLDQAISLLSSASSQVKLGSLQQARYDARIDQLRQLQERFKPYTKM |

**Figure 3 Sample Request email**

So that the administrator could keep track of the current server statues and prevent too much workload on the server.

After the server app notified the relevant parties, it will execute the following program:

1. Alignment generator
2. Core-algorithm helper

 After process are done, the program will send the result back to the email address provided by task submitter (e.g. models@predictioncenter.org)



**Figure 4 Sample result email**

* 1. **Alignment generator**

It was a shared module with MUfold written in C++. The program performs following steps:

1. Executes SSPro [23] on the input sequence to predict secondary structure.
2. Executes PSIPred [24] to predict secondary structure.
3. Executes BLAST (blastpgp -C, blastpgp -R, blastpgp -Q) to generate alignments.
4. Execute HHsearch to generate alignments.
5. Parse the result in to JSON format and retrieve the 3D coordinates of the alignments from the protein database.



**Figure 5 Raw blast results**



**Figure 6 Parsed Blast result**

* 1. **Core-algorithm helper**

This program is written in Go. It does not compute prediction information itself, but provides the following functions:

1. Provide an interface to the core algorithm that is compatible with other MUfold programs.
2. Check whether input parameters are valid
3. Check if all paths are accessible
4. Check is GDT-TS scoring server is running
5. Execute and monitor the MUfoldQA core program
6. Send brief report when program is finished and report errors if there is there is any.



**Figure 7 Sample report message**

* 1. **MUfoldQA core program**

This part was written in MATLAB. It is the actual executer of the algorithm.

* + 1. **Main module**

When Core-algorithm helper calls this module, it will pass the target name, decoy directory, template directory and output directory to the later. Then the main module enters the template directory and call template JSON loader to read the template information into a template database. Then it calls stage 1 alignment generator to select top templates from the template database, trim them, and calculate per c-alpha position weight for each selected alignment. The results are then stored in stage 1 alignment set and stage 1 weight set respectively. After that, the module enters the decoy directly. For each decoy, the module passes the decoy file path, stage 1 alignment set and stage 1 weight set to the stage 1 score calculator to compute both global score and per c-alpha position local score. Both global and local scores are stored into stage 2 reference model database while the stage 1 global score is directly outputted as the score of MUfoldQA\_S. Then the module will execute the stage 2 reference model generator. The later will read decoy information from pdb files. And based on information from stage 2 reference model database, it will generate stage 2 reference model set and stage 2 weight set. Then, for each decoy, it will pass the decoy file path, stage 2 reference model set and stage 2 weight set to the stage 1 score calculator and the result will be used as the output of MUfoldQA\_S

* + 1. **Template JSON loader**

The template JSON loader is a function that uses regular expression to read the JSON file generated by alignment generator and score it to a template database. The database contains the following fields:

|  |  |  |
| --- | --- | --- |
| Name | Type | Caption |
| Target Name | String | The name of the target |
| Target Full Sequence | String | The entire sequence of the target |
| Template Name | String | The name of the template in which we found the matching alignment |
| Query Start | Integer | The beginning position on target that matches the alignment |
| Query End | Integer | The ending position on target that matches the alignment |
| Target Length | Integer | Length of the target sequence |
| Score | Double | The BitScore of alignment |
| Expect | Double | The E-value of the alignment |
| Identities | Double | The ratio of the length of the sequence that exactly matches to the length of the entire match |
| Query Part | String | The matching part of the target sequence |
| Subject Part | String | The matching part of the template sequence (Also known as alignment sequence) |
| Original Index | Integer | The original rank in the output file |
| File Name | String | The name of the file that contains 3D coordinates |
| Cover Rate | Double | The ration of length of the none gap sequence to the length of the entire alignment sequence |
| Sort Score | Double | An artificial score used to sort all the alignments |
| Alignment Structure | Array / Double | Stores the 3D coordinates of the alignment |
| Individual Weights | Array / Double | Stores the weight of each Amino Acid |

**Table 1 Template database data structure**

Among these fields, cover rate sort score was calculated as $(3-log\_{10}E)∙I∙C$. In which *E* is Expect, *I* is Identities and *C* is Cover Rate.

Because sometimes the Blast or HHsearch could find the template, but the 3D coordinates of template is not available. Every time before the program record a specific template into the database, it will check if the corresponding coordinate file exists and only add the template to the database if it does.

* + 1. **Stage 1 alignment generator**

This module will first retrieve all the templates of a specific target from the database and then sort them in descending order based on the field Sort Score. Then the program will go through each templates and load coordinates from the file. If there is a gap in the file, it will be marked as NaN. Then the program read Query Start and Query End field to determine the sequential relative position of the template to the target. Based on such information the program pads the head and tail of the template to make it of the same length as the target. This result is saved in stage 1 alignment set. Then, it retrieves the matching part of the template sequence and the target sequence. For each c-alpha position, it retrieves the corresponding value B from the BLOSUM45 lookup table and set weight=$2^{B}$. After the weight is calculated for the entire sequence, the program add padding to the head and tail of the sequence to make sure the weight value is corresponding to the coordinates. After all is done, the result is stored in stage 1 weight set using the same index number as the corresponding structure using on the stage 1 alignment set.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | B | J | Z | X | \* |
| A | 5 | -2 | -1 | -2 | -1 | -1 | -1 | 0 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 1 | 0 | -2 | -2 | 0 | -1 | -1 | -1 | -1 | -5 |
| R | -2 | 7 | 0 | -1 | -3 | 1 | 0 | -2 | 0 | -3 | -2 | 3 | -1 | -2 | -2 | -1 | -1 | -2 | -1 | -2 | -1 | -3 | 1 | -1 | -5 |
| N | -1 | 0 | 6 | 2 | -2 | 0 | 0 | 0 | 1 | -2 | -3 | 0 | -2 | -2 | -2 | 1 | 0 | -4 | -2 | -3 | 5 | -3 | 0 | -1 | -5 |
| D | -2 | -1 | 2 | 7 | -3 | 0 | 2 | -1 | 0 | -4 | -3 | 0 | -3 | -4 | -1 | 0 | -1 | -4 | -2 | -3 | 6 | -3 | 1 | -1 | -5 |
| C | -1 | -3 | -2 | -3 | 12 | -3 | -3 | -3 | -3 | -3 | -2 | -3 | -2 | -2 | -4 | -1 | -1 | -5 | -3 | -1 | -2 | -2 | -3 | -1 | -5 |
| Q | -1 | 1 | 0 | 0 | -3 | 6 | 2 | -2 | 1 | -2 | -2 | 1 | 0 | -4 | -1 | 0 | -1 | -2 | -1 | -3 | 0 | -2 | 4 | -1 | -5 |
| E | -1 | 0 | 0 | 2 | -3 | 2 | 6 | -2 | 0 | -3 | -2 | 1 | -2 | -3 | 0 | 0 | -1 | -3 | -2 | -3 | 1 | -3 | 5 | -1 | -5 |
| G | 0 | -2 | 0 | -1 | -3 | -2 | -2 | 7 | -2 | -4 | -3 | -2 | -2 | -3 | -2 | 0 | -2 | -2 | -3 | -3 | -1 | -4 | -2 | -1 | -5 |
| H | -2 | 0 | 1 | 0 | -3 | 1 | 0 | -2 | 10 | -3 | -2 | -1 | 0 | -2 | -2 | -1 | -2 | -3 | 2 | -3 | 0 | -2 | 0 | -1 | -5 |
| I | -1 | -3 | -2 | -4 | -3 | -2 | -3 | -4 | -3 | 5 | 2 | -3 | 2 | 0 | -2 | -2 | -1 | -2 | 0 | 3 | -3 | 4 | -3 | -1 | -5 |
| L | -1 | -2 | -3 | -3 | -2 | -2 | -2 | -3 | -2 | 2 | 5 | -3 | 2 | 1 | -3 | -3 | -1 | -2 | 0 | 1 | -3 | 4 | -2 | -1 | -5 |
| K | -1 | 3 | 0 | 0 | -3 | 1 | 1 | -2 | -1 | -3 | -3 | 5 | -1 | -3 | -1 | -1 | -1 | -2 | -1 | -2 | 0 | -3 | 1 | -1 | -5 |
| M | -1 | -1 | -2 | -3 | -2 | 0 | -2 | -2 | 0 | 2 | 2 | -1 | 6 | 0 | -2 | -2 | -1 | -2 | 0 | 1 | -2 | 2 | -1 | -1 | -5 |
| F | -2 | -2 | -2 | -4 | -2 | -4 | -3 | -3 | -2 | 0 | 1 | -3 | 0 | 8 | -3 | -2 | -1 | 1 | 3 | 0 | -3 | 1 | -3 | -1 | -5 |
| P | -1 | -2 | -2 | -1 | -4 | -1 | 0 | -2 | -2 | -2 | -3 | -1 | -2 | -3 | 9 | -1 | -1 | -3 | -3 | -3 | -2 | -3 | -1 | -1 | -5 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | 0 | 0 | -1 | -2 | -3 | -1 | -2 | -2 | -1 | 4 | 2 | -4 | -2 | -1 | 0 | -2 | 0 | -1 | -5 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -1 | -1 | 2 | 5 | -3 | -1 | 0 | 0 | -1 | -1 | -1 | -5 |
| W | -2 | -2 | -4 | -4 | -5 | -2 | -3 | -2 | -3 | -2 | -2 | -2 | -2 | 1 | -3 | -4 | -3 | 15 | 3 | -3 | -4 | -2 | -2 | -1 | -5 |
| Y | -2 | -1 | -2 | -2 | -3 | -1 | -2 | -3 | 2 | 0 | 0 | -1 | 0 | 3 | -3 | -2 | -1 | 3 | 8 | -1 | -2 | 0 | -2 | -1 | -5 |
| V | 0 | -2 | -3 | -3 | -1 | -3 | -3 | -3 | -3 | 3 | 1 | -2 | 1 | 0 | -3 | -1 | 0 | -3 | -1 | 5 | -3 | 2 | -3 | -1 | -5 |
| B | -1 | -1 | 5 | 6 | -2 | 0 | 1 | -1 | 0 | -3 | -3 | 0 | -2 | -3 | -2 | 0 | 0 | -4 | -2 | -3 | 5 | -3 | 1 | -1 | -5 |
| J | -1 | -3 | -3 | -3 | -2 | -2 | -3 | -4 | -2 | 4 | 4 | -3 | 2 | 1 | -3 | -2 | -1 | -2 | 0 | 2 | -3 | 4 | -2 | -1 | -5 |
| Z | -1 | 1 | 0 | 1 | -3 | 4 | 5 | -2 | 0 | -3 | -2 | 1 | -1 | -3 | -1 | 0 | -1 | -2 | -2 | -3 | 1 | -2 | 5 | -1 | -5 |
| X | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -5 |
| \* | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | 1 |

**Table 2 Weight Lookup Table**

* + 1. **Stage 1 score calculator**

This functions take in the decoy path, stage 1 alignment set and stage 1 weight set and give scores for each c-alpha position as well as a global score. First, it checks all alignments in stage 1 alignment set to find the max length, then pass the max length and the decoy path to advanced decoy loader to retrieve the 3D coordinates of the model to be scored. Based on the loaded decoy data, the program calculates a mask of which c-alpha position has valid value. Then, for each alignment in the stage 1 alignment set, the program also calculates a mask of which c-alpha position has both valid 3D coordinates and weights. Then the common part of two masks are calculated for a global mask. The global mask is then applied to both decoy and alignment 3D structures to extract the overlapping region: overlapped decoy and overlapped alignment. The program encodes the 3D structure information of the latter two and send a request to the GDT-TS calculator which then return the GDT-TS value reflecting the homogeneity between these two fragments. After a similar score is calculated for all the alignments, the program goes through each c-alpha position, and calculate a weighted average of all these scores weighted by the value of corresponding c-alpha position on corresponding weight matrix as the overall score on this c-alpha position. The score of all the position with no valid value is set to zero. This overall score matrix is stored in stage 1 score matrix set. Finally, this function computes the average of the score matrix and store the result in stage 1 score set which is then outputted as the final score of MUfoldQA\_S.

* + 1. **Stage 2 reference model generator**

Compared with stage 1 score calculator, the operation of stage 2 reference model generator is relatively simple. It calls advanced decoy loader to retrieve the 3D coordinates. Then it stores the structure to stage 2 reference model set and corresponding stage 1 score matrix to stage 2 weight set in the descending order of the stage 1 score.

* + 1. **Stage 2 score calculator**

This functions take in the decoy path, stage 2 reference model set and stage 2 weight set and give scores for each c-alpha position as well as a global score. First, it checks all reference models in stage 2 reference model set to find the max length, then pass the max length and the decoy path to advanced decoy loader to retrieve the 3D coordinates of the model to be scored. Based on the loaded decoy data, the program calculates a mask of which c-alpha position has valid value. Then, for each reference model in the stage 2 reference model set, the program also calculates a mask of which c-alpha position has both valid 3D coordinates and weights. Then the common part of two masks are calculated for a global mask. The global mask is then applied to both decoy and reference model 3D structures to extract the overlapping region: overlapped decoy and overlapped reference model. The program encodes the 3D structure information of the latter two and send a request to the GDT-TS calculator which then return the GDT-TS value reflecting the homogeneity between these two fragments. After a similar score is calculated for all the reference model, the program goes through each c-alpha position, and calculate a weighted average of all these scores weighted by the value of corresponding c-alpha position on corresponding weight matrix as the overall score on this c-alpha position. The score of all the position with no valid value is set to zero. This overall score matrix is stored in stage 2 score matrix set. Finally, this function computes the average of the score matrix and store the result in stage 2 score set which is then outputted as the final score of MUfoldQA\_C.

* + 1. **advanced decoy loader**

To save time, the program initially assumes the coordinates in files are of the correct order. It scans PDB file line by line, if the length of the line is larger than 5, then it checks if the initial of the line is ‘ATOM’. If true, then it checks if the character 13-16 is CA which is the indicator of c-alpha. If it is also true, then it records the character 7-11 to SerialNo, 23-26 to CA, 31-28,39-46, 47-54, to X, Y, Z coordinates respectively. If the difference between the current and previous SerialNo is larger than one, then it will pad all the SerialNo in between. After the program load in the PDB file, it will check if the length of the loaded decoy is the same as expected value, if it is, then it will directly return the X, Y, Z. If it is not, this means the assumption that the PDB file being in right order is wrong, then is will create an empty array of the expected decoy length, and then fill the X, Y, Z coordinates data to the array based on the SerialNo.

* 1. **GDT-TS calculator**

This part was originally implemented in MATLAB, but due to performance issues, it was then re-implemented in Go. The module uses server-client architecture, and follows RESTFUL design principle. The server program is a multi-threaded and is capable of automatically detect the number of cores the server has and adjust parallelization accordingly. The client serializes the x, y, z coordinates of both proteins and send the request to server. The server first verifies if the request is valid, then extract the 3D structures of both proteins from the request. To calculate GDT-TS score, a modifies version of TMscore [25, 26] was used. The major advantage of the modified version over the original version is that the former does not need a PDB file as the input. Such modification saves a lot of time since it avoids both generating and reading PDB files. Most importantly, the GDT-TS calculation is being executed excessive amount of times, by avoiding I/O from SSD/HDD completely, such modification is beneficial for the overall life expectancy of the hard drive and the performance of the system. After the GDT-TS calculation is done, the server will return the following results to the client:

RMSD, TM-score, MaxSub-score, GDT-TS, GDT\_HA, structure A Length, structure B Length.



**Figure 8 Sample HTTP traffic**

1. **EXPERIMENTS AND RESULTS**

To test the performance of both algorithm, we tested our algorithm on decoys from CASP 11. To compare the result with other algorithms, we directly used the official CASP results from its official website.

* 1. **Data Set**
		1. **Targets**

At the time of experiment, we noticed that decoys for some of the targets are not available for download on the official FTP. Also, on the automatic evaluation section on CASP website, only results for certain targets are accessible. So, based on the consideration of both aspect, we used following list of targets from CASP 11 for out evaluation.

|  |
| --- |
| T0759, T0760, T0761, T0762, T0763, T0764, T0765, T0766, T0767, T0768, T0769, T0770, T0771, T0772, T0773, T0774, T0776, T0777, T0780, T0781, T0782, T0783, T0784, T0785, T0786, T0787, T0788, T0789, T0790, T0792, T0794, T0796, T0800, T0801, T0803, T0805, T0806, T0807, T0808, T0810, T0811, T0812, T0813, T0815, T0816, T0817, T0818, T0819, T0821, T0822, T0823, T0824, T0827, T0829, T0830, T0831, T0832, T0833, T0834, T0835, T0836, T0837, T0838, T0840, T0841, T0843, T0845, T0847, T0848, T0849, T0851, T0852, T0853, T0854, T0855, T0856, T0858 |

* + 1. **Server models**

For each targets, the algorithms are tested in two different stages. According to the rules, the first stage selects up to 20 models that covers the whole range of models and the second stage selects about 150 top models based on the ranking from the naïve consensus QA method. It should be noted that the stage 2 data set is only released after the stage 1 quality assessment submission dead line, so the decoys from stage 2 cannot be use used to evaluate the decoys in stage 1.

Among all the targets we used, all of them provide 20 decoys in stage 1. And most of them provides 150 decoys in stage 2, except T0808, T0761, T0827 comes with 160 decoys, T0831, T0834 data set contains 155 decoys and 154 decoys for target T0763.

* 1. **Experiment procedure**
		1. **Speed-up test**

We run the program on CASP 11 targets, first time use the MATLAB version of GDT-TS calculator, the second time use Golang version. The rest of the program are both using the same MATLAB implementation. Alignments for each targets are provided. So the running time will be strictly the algorithm itself. Record the time used for each target.

* + 1. **Accuracy test**
1. For each target, we downloaded the PDB file of native structure as well as the decoys from the official CASP website.
2. For each decoy, compare it to the native structure and calculate GDT-TS score (true GDT\_TS score)
3. For each decoy, calculate scores using both MUfoldQA\_S and MUfoldQA\_C algorithm
4. For each target calculate Pearson Correlation Coefficient between the score given by the algorithm and the true GDT-TS score.
	1. **Evaluation Parameters**

Pearson Correlation Coefficient is calculated as following:

$$r =\frac{\sum\_{i=1}^{n}(X\_{i}-\overbar{X})(Y\_{i}-\overbar{Y})}{\sqrt{\sum\_{i=1}^{n}(X\_{i}-\overbar{X})^{2}}\sqrt{\sum\_{i=1}^{n}(Y\_{i}-\overbar{Y})^{2}}}$$

In which *X* is predicted score, *Y* is true GDT-TS score, *n* is the number of the decoys for the specific target and *i* is the index of the decoy.

In general, the higher Pearson Correlation Coefficient is, the more accurate is the predicted score considered to be.

* 1. **Collecting data for other algorithms**

The Pearson Correlation Coefficient of other algorithms are downloaded from the “automatic evaluation” section of CASP 11 official website (available at <http://www.predictioncenter.org/casp11/qa_analysis.cgi>). If a team failed to submit the result for some targets, the Pearson Correlation Coefficient will be set to zero for that team on the target.

* 1. **Results and analysis**

We compared the result of our algorithm with other algorithms that also participated in CASP 11 competition on both stage 1 and stage 2 dataset. And ranked them based on the average of Pearson Correlation Coefficient to real GDT-TS score.

* + 1. **Speedup test**

**Figure 9 Speed comparison**

The figure shows that the max speed up is 6.2 times and even min speed up could reach 1.9 times. With the average is 4.3 times, we can see that even though the HTTP traffic caused extra overhead time, the GO version of GDT-TS calculator is much faster than its MATLAB equivalent.

* + 1. **Overall ranking**
			1. **Stage 1 ranking**

|  |  |  |
| --- | --- | --- |
| Group Name | Submission Count | Average Pearson Correlation |
| DAVIS-QAconsensusALL | 77 | 0.8519 |
| MUfoldQA\_C | 77 | 0.8458 |
| MUfoldQA\_S | 77 | 0.8157 |
| MULTICOM-REFINE | 77 | 0.8139 |
| Pcons-net | 77 | 0.8106 |
| DAVIS-QAconsensus | 77 | 0.8083 |
| MUFOLD-QA | 77 | 0.8076 |
| MUFOLD-Server | 77 | 0.8055 |
| nns | 77 | 0.7854 |
| MQAPsingleA | 71 | 0.7793 |
| Wallner | 77 | 0.7764 |
| MQAPmulti | 71 | 0.7522 |
| ModFOLDclust2 | 77 | 0.7426 |
| MQAPsingle | 71 | 0.7418 |
| ModFOLD5 | 77 | 0.7406 |
| ModFOLD5\_single | 77 | 0.7389 |
| ConsMQAPsingle | 71 | 0.7198 |
| MULTICOM-CONSTRUCT | 77 | 0.6811 |
| MQAPsingleB | 71 | 0.6797 |
| Wang\_SVM | 77 | 0.6722 |
| ProQ2-refine | 77 | 0.6698 |
| ProQ2 | 77 | 0.6589 |
| myprotein-me | 76 | 0.6547 |
| MULTICOM-CLUSTER | 77 | 0.6530 |
| Wang\_deep\_2 | 77 | 0.6484 |
| MULTICOM-NOVEL | 77 | 0.6467 |
| Wang\_deep\_3 | 77 | 0.6425 |
| PconsD | 75 | 0.6411 |
| Wang\_deep\_1 | 77 | 0.6313 |
| BITS | 77 | 0.6271 |
| RFMQA | 77 | 0.6189 |
| VoroMQA | 77 | 0.5681 |
| keasar | 77 | 0.5598 |
| raghavagps-qaspro | 77 | 0.3624 |
| Qpotclust | 31 | 0.2831 |
| LNCCUnB | 54 | 0.2790 |
| Qpotfilt | 31 | 0.2712 |
| Qpot | 31 | 0.2274 |
| MUFOLD-DQA | 17 | 0.1866 |
| FUSION | 77 | 0.0784 |
| OccuScore | 77 | 0.0000 |
| DandekarLab | 26 | -0.0033 |

**Table 3 Ranking of CASP 11 stage 1 dataset**

It need to be noted that the group “DAVIS-QAconsensusALL” is actually naïve consensus algorithm with a twist that it uses all the models been submitted as reference model instead of the selected 20. It is not using the same amount of information like everyone else. Thus it is only used as a benchmark instead of an actual valid algorithm.

Based on the chart above, we can see that on stage 1 dataset, the MUfoldQA\_C ranks the best QA algorithm among all groups with a 0.8458 average Pearson Correlation and beat the second by a significant 0.03. And MUfoldQA\_S ranks the second place and is outperforming most of other algorithms with a very huge lead.

* + - 1. **Stage 2 ranking**

|  |  |  |
| --- | --- | --- |
| Group Name | Sample Size | Normalized |
| Pcons-net | 77 | 0.6484 |
| Wallner | 77 | 0.6417 |
| MUfoldQA\_C | 77 | 0.5819 |
| MUFOLD-Server | 77 | 0.5681 |
| DAVIS-QAconsensusALL | 77 | 0.5613 |
| DAVIS-QAconsensus | 77 | 0.5550 |
| MULTICOM-REFINE | 77 | 0.5538 |
| ModFOLDclust2 | 77 | 0.5488 |
| MUFOLD-QA | 76 | 0.5463 |
| MULTICOM-CONSTRUCT | 77 | 0.5404 |
| nns | 77 | 0.5305 |
| MQAPsingleA | 66 | 0.5019 |
| PconsD | 75 | 0.4899 |
| ModFOLD5 | 77 | 0.4852 |
| MUfoldQA\_S | 77 | 0.4758 |
| MQAPmulti | 67 | 0.4556 |
| ConsMQAPsingle | 67 | 0.4429 |
| MQAPsingle | 67 | 0.4237 |
| MULTICOM-CLUSTER | 77 | 0.4170 |
| VoroMQA | 77 | 0.4142 |
| myprotein-me | 76 | 0.4100 |
| MULTICOM-NOVEL | 77 | 0.4056 |
| ModFOLD5\_single | 77 | 0.4040 |
| ProQ2-refine | 77 | 0.3835 |
| ProQ2 | 77 | 0.3827 |
| Wang\_SVM | 77 | 0.3779 |
| MQAPsingleB | 67 | 0.3692 |
| RFMQA | 76 | 0.3645 |
| BITS | 77 | 0.3172 |
| Wang\_deep\_2 | 77 | 0.3157 |
| Wang\_deep\_3 | 77 | 0.3098 |
| Wang\_deep\_1 | 77 | 0.3091 |
| keasar | 72 | 0.2983 |
| Qpotclust | 30 | 0.2926 |
| raghavagps-qaspro | 77 | 0.2393 |
| Qpotfilt | 26 | 0.2039 |
| Qpot | 30 | 0.1681 |
| LNCCUnB | 58 | 0.0890 |
| MUFOLD-DQA | 14 | 0.0810 |
| DandekarLab | 27 | 0.0590 |
| FUSION | 77 | 0.0521 |
| OccuScore | 77 | 0.0000 |

**Table 4 Ranking of CASP 11 stage 2 dataset**

Same as the stage 1, “DAVIS-QAconsensusALL” is still using the entire decoy pool instead of the stage 2 set like all the others. However, the stage 2 set is a much larger subset of the decoy pool compared with stage 1 set. The advantage of “DAVIS-QAconsensusALL” has been significantly reduced. Thus, in this set, we can observe that groups like Pcons-net [27], Wallner, MUfoldQA\_C and MUFOLD-Server is performing better than the benchmark group. It could be also noted that when the dataset is sufficiently large, a well-designed consensus based algorithm generally performs better than single or quasi-single model QA methods. Among all the methods, the MUfoldQA\_C ranks the third after Pcons-net and Wallner. Between all the single and quasi-single model QA methods, MUfoldQA\_S ranks the third after nns, MQAPsingleA.

* + 1. **Comparison between different methods**
			1. **Comparison between MUfoldQA\_S and MUfoldQA\_C**



**Figure 10 Stage 1 Per target correlation comparison between MUfoldQA\_C and MUfoldQA\_S**

From the figure of stage 1 we can see that when tested on the stage 1 dataset both MUfoldQA\_S and MUfoldQA\_C will give a score with very high correlation to the true GDT-TS even though they are two very different types of algorithm.

Based on the raw data, we could also observe that for the cases that is easy or hard for both algorithm, both of them will perform very similarly. However, in the case of medium difficulty, the two algorithms will perform very differently, and it can be easily noticed that the MUfoldQA\_C actually performs much better in the medium cases.

The reason behind this, is that the MUfoldQA\_C is actually the fusion of two distinct algorithms, the MUfoldQA\_S and Naïve consensus. If the target is hard (in terms of the actually difficulty of structure prediction), the Blast and HHsearch would be very unlikely able to find structures that are similar to the true structure of the target. When MUfoldQA\_S uses these structures as the templates to score the decoys, the result will suffer. While the score given by MUfoldQA\_C is more leaning towards the naïve consensus score, it is not immune from such problem. Due to low template quality, the generated models will also be likely bad. And when these models been used as the reference, the result will suffer as well. So in case of hard targets, both algorithms may perform uniformly bad. For the medium targets. The average quality of the predicted model may not be uniformly consistent with the quality of the templates. The weighting process may mitigate the problem of the decoy quality or make it even worse. So the results may turn out both ways. However, when the targets are easy, the quality of both templates and decoys will have relatively high chance to be good. In this case, both algorithms will perform very well and results may be very similar between each other.



**Figure 11 Stage 2 Per target correlation comparison between MUfoldQA\_C and MUfoldQA\_S**

However, on stage 2 dataset, the performance of MUfoldQA\_C is significantly better than MUfoldQA\_S. This is because in the stage 1, the number of decoys been provided is only 20, which is far too small compare with stage 2 dataset containing about 150 decoys for each targets. More importantly, the stage 1 decoys are covering the entire range of the decoys from good to poor but stage 2 decoys are from the top ones. This difference helps the consensus based algorithms to get higher quality of scores but of little help to single or quasi-single model QA methods which creates an ineligible advantage for the former.

* + - 1. **Comparison between MUfoldQA\_S and other quasi-single model QA methods**
				1. **Comparison between MUfoldQA\_S and MQAPsingle\_A**



**Figure 12 Stage 1 Per target correlation comparison between MUfoldQA\_S and MQAPsingle\_A**

MQAPsingle\_A is a quasi-single QA method that utilizes GeneSilico Fold prediction meta-server to generate about a hundred 3D models based on the target sequence, then the program uses these models as reference model to calculate the GDT-TS score of the decoys. The final score is the average of all the score. The major difference between MQAPsingle\_A and MUfoldQA\_S is that the former actually needs to generates complete models while the later could directly make use of the raw fragments found by Blast and HHsearch programs. The MQAPsingle\_A program clearly performs better on medium targets but performs significantly worse on the easy ones.



**Figure 13 Stage 2 Per target correlation comparison between MUfoldQA\_S and MQAPsingle\_A**

The result on the stage 2 dataset is quite different from stage 1. In this one, we could see that in most cases, the MQAPsingle\_A performs better than MUfoldQA\_S. Judging from the raw data, it can be seen that the former reaches very high correlation with true GDT-TS on most of easy and medium targets but the later could not achieve high correlation on some cases most of others do. This in turn makes the average correlation relatively lower for the MUfoldQA\_S.

* + - * 1. **Comparison between MUfoldQA\_S and ModFOLD5\_single**



**Figure 14 Stage 1 Per target correlation comparison between MUfoldQA\_S and ModFOLD5\_single**

ModFOLD5\_single [28] is a quasi-single model QA method. It uses the models from IntFOLD3 [28] server as reference models and calculate the in a similar manner of ModFOLDclust2 [29]. From the figure, we can see that MUfoldQA\_S is performing better than ModFOLD5\_single in most cases no matter if the target is easy, medium or hard. Especially in the medium cases, the score given by ModFOLD5\_single might even exhibit negative correlation with the true GDT-TS, but the MUfoldQA\_S would remain strongly positive. For example, in the case of targets: T0777, T0789 and T0832. The global average GDT-TS correlation are 0.6419, 0.4792 and 0.5791, respectively. The ModFOLD5\_single only achieved -0.109, -0.187 and -0.103. But MUfoldQA\_S was able to score well-above-average correlations like 0.8185, 0.6334 and 0.6596, respectively. And that was an astonishing 0.9275, 0.8204 and0.7626 better than ModFOLD5\_single.



**Figure 15 Stage 2 Per target correlation comparison between MUfoldQA\_S and ModFOLD5\_single**

On the stage 2 dataset, the overall result is very similar. For most cases, the MUfoldQA\_S is performing better than ModFOLD5\_single. Also, out of 77 targets, the scores calculated by MUfoldQA\_S have negative correlation to true GDT-TS on 11 target, while for ModFOLD5\_single that number is 19. The negative correlation is dangerous because it gives a partial or completely reverse ranking for the models When about a quarter of the targets is given a revers ranking, the credibility of an algorithm becomes very questionable.

* + - * 1. **Comparison between MUfoldQA\_S and myprotein-me**



**Figure 16 Stage 1 Per target correlation comparison between MUfoldQA\_S and myprotein-me HHH**

myprotein-me is a very interesting method to be compared with. It calculates features like contacts satisfaction, secondary structure information from PSIPRED, as well as statistical potentials from dDFIRE , RW/RW+ [13] and ORDER\_AVE [30]. Then they used pre-trained random forest to combine the features and generate final score. From the stage 1 result. It can be clearly seen that the MUfoldQA\_S is dominating this dataset. Out of 77 targets, it gets higher correlation with true GDT-TS on 61 of them. Also, myprotein-me get negative correlation on 3 targets while for MUfoldQA\_S, all of the correlations are positive.



**Figure 17 Stage 2 Per target correlation comparison between MUfoldQA\_S and myprotein-me**

And on the stage 2 dataset, such trend continues. MUfoldQA\_S outperformed myprotein-me on 49 targets and achieved average correlation 0.4758 while myprotein-me only achieved 0.4100. Also, we can see that both algorithm cannot get as good result as it did in the stage1, but MUfoldQA\_S still has distinct advantage no matter the target is easy medium or hard. It has to be pointed out that myprotein-me only get negative correlation on 4 targets while that number for MUfoldQA\_S is 11. However, all things considered, especially huge advantage of 0.0658 in average true GDT-TS correlation, MUfoldQA\_S is still considered as a better algorithm.

* + - 1. **Comparison between MUfoldQA\_C and other multi-model QA methods**
				1. **Comparison between MUfoldQA\_C and Pcons-net**



**Figure 18 Stage 1 Per target correlation comparison between MUfoldQA\_C and Pcons-net**

In CASP 11 Pcons-net was also a consensus based QA algorithm [27]. When applies both MUfoldQA\_C and Pcons-net to the stage 1 dataset, we can see the former performs much better. Out of 77 targets, the MUfoldQA\_C outperforms Pcons-net on 48 of them. And MUfoldQA\_C also beat Pcons-net on average true GDT-TS correlation by 0.0352. That been said, both algorithms are very well-designed algorithms as none of them showed negative correlation on stage 1 dataset.



**Figure 19 Stage 2 Per target correlation comparison between MUfoldQA\_C and Pcons-net**

However, the situation is much reversed on the stage 2 dataset. Out of 77 targets, the Pcons-net outperforms MUfoldQA\_C on 47 of them. And Pcons-net also beat MUfoldQA\_C on average true GDT-TS correlation by 0.0665. As for the negative correlation, it only has been observed on 2 and 7 targets. It is quite good results all things considered.

* + - * 1. **Comparison between MUfoldQA\_C and MULTICOM-REFINE**



**Figure 20 Stage 1 Per target correlation comparison between MUfoldQA\_C and MULTICOM-REFINE**

The group MULTICOM-REFINE employs a very interesting strategy. It first decides whether the target is easy or hard. If it is relatively easy, it utilizes APOLLO [31] which to calculate a pair-wise GDT-TS between the decoy and all other models in the dataset as well as the models in the model pool. Then it calculates the average as the final output. If the target is considered hard, then it will use ModelEvaluator [32] which is a single model QA method that considers features like secondary structure, relative solvent accessibility, contact map, and beta sheet structure. All these features are then combined by machine learning method to give a final score. From the stage 1 results. It is interesting there is some correlation (0.88) between the performance of MUfoldQA\_C and MULTICOM-REFINE. However, for most of the targets, the MUfoldQA\_C algorithm outperforms the MULTICOM-REFINE.

****

**Figure 21 Stage 2 Per target correlation comparison between MUfoldQA\_C and MULTICOM-REFINE**

On the Stage two result, we observed a very similar strong correlation (0.85) between the performance of MUfoldQA\_C and MULTICOM-REFINE, especially for the simple targets. It is understandable since the former is basically a very carefully weighted consensus and the latter is a consensus with a larger pool. The correlation becomes weaker when the targets are relatively hard because for these targets the MULTICOM-REFINE will be very likely to use the single model QA instead of consensus. However, it is very hard to predict the difficulty level for each target accurately before knowing the final result. It is possible to misjudge the situation and still use consensus based algorithm in the actual hard case. So we could still observe some correlation between the results of these two even when the targets are hard.

* + - * 1. **Comparison between MUfoldQA\_C and DAVIS-QAconsensus**



**Figure 22 Stage 1 Per target correlation comparison between MUfoldQA\_C and DAVIS-QAconsensus**

As we have mentioned above, the MUfoldQA\_C is basically a very carefully per c-alpha position weighted consensus. It would be very interesting to compare it with naïve consensus and see how much performance gain did we achieve through this weighting operation. So, we compare the result with DAVIS-QAconsensus which is a benchmark server running naïve consensus algorithm. As expected, the performance of MUfoldQA\_C and DAVIS-QAconsensus shows a very strong correlation of 0.82. Also, out of 77 targets, the MUfoldQA\_C gets better results on 49 (64%) of them while loss very little on the rest 28. More interestingly, the target with negative correlation -0.479 when scored with naïve consensus has been reversed to positive correlation 0.499 with true GDT-TS, with a correlation gain of 0.978. The new weighted consensus is much better than the original one.



**Figure 23 Stage 2 Per target correlation comparison between MUfoldQA\_C and DAVIS-QAconsensus**

The similar situation can also be seen with the stage 2 results. The performance of two scores are highly correlated with the correlation of 0.85. It can be seen that MUfoldQA\_C again has much advantage in the medium targets.

If we compare the two graphs for stage 1 and stage 2, we can see that although MUfoldQA\_C outperforms naïve consensus in most cases, the main improvement is achieved for MUfoldQA\_C is when the target is hard and reference pool size is small. On such situation, the DAVIS-QAconsensus will get many poor consensuses because the reference model quality is poor, since the pool size is small, the chance for relatively better consensus to stand out is also very low. However, with the weighting scheme, the program could better distinguish the good one from the bad and assign different weight accordingly. And this is the reason why we can see very high improvement for such cases.

* + 1. **Robustness comparison**

**Figure 24 Ranking of stage 1 and stage 2 sorted by average of both**

To evaluate the overall performance of all the algorithms, we calculated the average ranking of stage 1 and stage 2. And then sorted them in ascending order. From the figure we can see MUfoldQA\_C has the best performance among all methods, and MUfoldQA\_S has the best performance among all single and quasi-single model QA methods.

1. **CONCLUSION AND FUTURE WORK**

In this thesis two fully automatic protein model quality assessment methods have been designed, implemented and tested the result shows that they are among the best of the QA methods. Especially the MUfoldQA\_C, ranked number one in stage one, number three in stage two, and also number one in average ranking. We could also see that both newly proposed algorithm perform extraordinarily well on the cases of hard targets or small-sized decoy pool, which is Achilles’s heel of ordinary consensus-based methods.

There are still many aspects could be further improved. Currently we are using a consensus size of up to 100, but we have observed that we could improve the correlation to true GDT-TS even more if we can find a way to dynamically predict the optimal consensus size. Also, we have come up with some ideas to make this method compute a local QA score in MUfoldQA\_C which might be quite useful in the refinement phase of the prediction.

**REFERENCES**

[1] H. Zhiquan, M. Wenji, J. Zhang, and D. Xu, “A new Hidden Markov Model for protein Quality Assessment using compatibility between protein sequence and structure,” *Tsinghua Science and Technology,* vol. 19, no. 6, pp. 559-567, 2014.

[2] L. Correa, B. Borguesan, C. Farfan, M. Inostroza-Ponta, and M. Dorn, “A Memetic Algorithm for 3-D Protein Structure Prediction Problem,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* vol. PP, no. 99, pp. 1-1, 2016.

[3] X. H. Hao, G. J. Zhang, X. G. Zhou, and X. F. Yu, “A Novel Method Using Abstract Convex Underestimation in Ab-Initio Protein Structure Prediction for Guiding Search in Conformational Feature Space,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* vol. 13, no. 5, pp. 887-900, 2016.

[4] G. Zhang, X. Yu, X. Zhou, and X. Hao, "A population-based conformational optimal algorithm using replica-exchange in ab-initio protein structure prediction." pp. 701-706.

[5] M. S. Johnson, N. Srinivasan, R. Sowdhamini, and T. L. Blundell, “Knowledge-Based Protein Modeling,” *Critical Reviews in Biochemistry and Molecular Biology,* vol. 29, no. 1, pp. 1-68, 1994/01/01, 1994.

[6] A. Chida, R. Harrison, and Y. Q. Zhang, "Protein model assessment using extented fuzzy decision tree with spatial neighborhood features." pp. 54-60.

[7] J. Xiaoyang, D. Qiwen, L. Xuan, and L. Bin, "Protein model quality assessment by learning-to-rank." pp. 91-96.

[8] P. L. M. Antczak, T. Ratajczak, J. Blazewicz, P. Lukasiak, and J. Blazewicz, "SphereGrinder - reference structure-based tool for quality assessment of protein structural models." pp. 665-668.

[9] A. Kryshtafovych, A. Barbato, K. Fidelis, B. Monastyrskyy, T. Schwede, and A. Tramontano, “Assessment of the assessment: Evaluation of the model quality estimates in CASP10,” *Proteins: Structure, Function, and Bioinformatics,* vol. 82, pp. 112-126, 2014.

[10] A. Kryshtafovych, K. Fidelis, and A. Tramontano, “Evaluation of model quality predictions in CASP9,” *Proteins: Structure, Function, and Bioinformatics,* vol. 79, no. S10, pp. 91-106, 2011.

[11] H. Zhou, and Y. Zhou, “Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction,” *Protein Science : A Publication of the Protein Society,* vol. 11, no. 11, pp. 2714-2726, 2002.

[12] M.-y. Shen, and A. Sali, “Statistical potential for assessment and prediction of protein structures,” *Protein Science : A Publication of the Protein Society,* vol. 15, no. 11, pp. 2507-2524, 2006.

[13] J. Zhang, and Y. Zhang, “A Novel Side-Chain Orientation Dependent Potential Derived from Random-Walk Reference State for Protein Fold Selection and Structure Prediction,” *PLoS ONE,* vol. 5, no. 10, pp. e15386, 2010.

[14] R. Samudrala, and J. Moult, “An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction,” *J Mol Biol,* vol. 275, no. 5, pp. 895-916, Feb 06, 1998.

[15] Y. Wu, M. Lu, M. Chen, J. Li, and J. Ma, “OPUS-Ca: A knowledge-based potential function requiring only Cα positions,” *Protein Sci,* vol. 16, no. 7, pp. 1449-63, Jul, 2007.

[16] A. Ray, E. Lindahl, and B. Wallner, “Improved model quality assessment using ProQ2,” *BMC Bioinformatics,* vol. 13, pp. 224, 2012.

[17] M. A. Kurowski, and J. M. Bujnicki, “GeneSilico protein structure prediction meta-server,” *Nucleic Acids Research,* vol. 31, no. 13, pp. 3305-3307, 2003.

[18] M. Pawlowski, L. Kozlowski, and A. Kloczkowski, “MQAPsingle: A quasi single-model approach for estimation of the quality of individual protein structure models,” *Proteins: Structure, Function, and Bioinformatics,* vol. 84, no. 8, pp. 1021-1028, 2016.

[19] S. Henikoff, and J. G. Henikoff, “Amino acid substitution matrices from protein blocks,” *Proceedings of the National Academy of Sciences of the United States of America,* vol. 89, no. 22, pp. 10915-10919, 1992.

[20] H. Zhou, and J. Skolnick, “Protein model quality assessment prediction by combining fragment comparisons and a consensus C(α) contact potential,” *Proteins,* vol. 71, no. 3, pp. 1211-1218, 2008.

[21] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” *Nucleic Acids Research,* vol. 25, no. 17, pp. 3389-3402, September 1, 1997, 1997.

[22] J. Söding, “Protein homology detection by HMM–HMM comparison,” *Bioinformatics,* vol. 21, no. 7, pp. 951-960, April 1, 2005, 2005.

[23] C. N. Magnan, and P. Baldi, “SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity,” *Bioinformatics,* vol. 30, no. 18, pp. 2592-2597, 2014.

[24] D. T. Jones, “Protein secondary structure prediction based on position-specific scoring matrices1,” *Journal of Molecular Biology,* vol. 292, no. 2, pp. 195-202, 9/17/, 1999.

[25] Y. Zhang, and J. Skolnick, “Scoring function for automated assessment of protein structure template quality,” *Proteins: Structure, Function, and Bioinformatics,* vol. 57, no. 4, pp. 702-710, 2004.

[26] J. Xu, and Y. Zhang, “How significant is a protein structure similarity with TM-score = 0.5?,” *Bioinformatics,* vol. 26, no. 7, pp. 889-895, 2010.

[27] B. Wallner, and A. Elofsson, “Identification of correct regions in protein models using structural, alignment, and consensus information,” *Protein Science : A Publication of the Protein Society,* vol. 15, no. 4, pp. 900-913, 2006.

[28] L. J. McGuffin, J. D. Atkins, B. R. Salehe, A. N. Shuid, and D. B. Roche, “IntFOLD: an integrated server for modelling protein structures and functions from amino acid sequences,” *Nucleic Acids Research,* vol. 43, no. Web Server issue, pp. W169-W173, 2015.

[29] L. J. McGuffin, and D. B. Roche, “Rapid model quality assessment for protein structure predictions using the comparison of multiple models without structural alignments,” *Bioinformatics,* vol. 26, no. 2, pp. 182-188, January 15, 2010, 2010.

[30] Y. Liu, J. Zeng, and H. Gong, “Improving the orientation-dependent statistical potential using a reference state,” *Proteins: Structure, Function, and Bioinformatics,* vol. 82, no. 10, pp. 2383-2393, 2014.

[31] P. Larsson, M. J. Skwark, B. Wallner, and A. Elofsson, “Assessment of global and local model quality in CASP8 using Pcons and ProQ,” *Proteins: Structure, Function, and Bioinformatics,* vol. 77, no. S9, pp. 167-172, 2009.

[32] Z. Wang, A. N. Tegge, and J. Cheng, “Evaluating the absolute quality of a single protein model using structural features and support vector machines,” *Proteins: Structure, Function, and Bioinformatics,* vol. 75, no. 3, pp. 638-647, 2009.