*Bioinformatics*, 2016, December

doi: 10.1093/bioinformatics/xxxxx

Advance Access Publication Date: DD Month YYYY

Manuscript Category

|  |
| --- |
| Subject Section  Comparative Protein Modelling  Sai Manikanta S Godavarthi, Deepika Joseph  Computer Science – EECS Department, Wichita State University, Wichita, Kansas  Abstract  **Motivation:**.  **Results:** With the given 10 proteins; 5 from CASP 11 and 5 from CASP 10 we have obtained better accuracy for 3 proteins comparing with the CASP results. Results for each of the protein are explained separately in the results section. We have also implemented a sample pipeline for automation of the process of modelling using modeller. Implemented few refinement steps on some proteins where the accuracy has been increased compared to the previous results before refinement.  **Contact:** sxgdavarthi@shockers.wichita.edu  **Supplementary information:** Supplementary data are available at *Bioinformatics* online. |

# Introduction

The term comparative protein modelling or homology modelling or template based modelling are referred as the same where our own main goal is to model a protein 3D structure using the know templates. Here the templates are those with similar sequence to that of our unknown query protein. Based on the known protein i.e. our template we will model the query protein using the properties and alignment of the known template. The template contains sufficient information of spatial arrangement of residues and internal structure which helps in predicting our model. Comparative modelling of protein sequence is more reliable than compared to that of ab initio methods as in the later, the model is entirely built using only the sequence rather depending on the template. Protein are one of most important functional units of our body, they do most of the work in cells and they are required for structure, function, and regulation of body’s tissues and organs. The three-dimensional structure of the protein determines the functionality of the protein. The four levels of proteins i.e. the primary structure which is a sequence of amino acid residues determine the peptide chain. In the secondary structure, hydrogen bonds between the amino acids creates alpha helix, which is a spiral or coiled molecule and pleated sheet, which looks like ribbon with regular peaks and valleys as a part of the fabric. The tertiary structure is for overall shape of the protein which are either globular or fibrous. Quaternary structures describe the proteins appearance.

Our method of protein modelling starts with taking a query sequence. Query sequence is the one which we want to model a three-dimensional structure for the protein then later identifying template and build model using the modeller tool and we validate our results using varies validation techniques available online and do structural analysis of the proteins using visualization tools.

# Methods

The whole process if protein modelling is described as follows:

1. Select the required protein and get the sequence.
2. Search for template in BLAST, PDB and other protein databases accordingly as required.
3. Find the better matching sequence for the query sequence, this is our template. We may have single or multiple templates. Align the template with our query.
4. Prepare our files in PIR format. Download the PDB formats for the templates. Save PIR formatted query file as .ali extension.
5. Start the modeller by giving input the query .ali file and other PDB files, according to the log file generated give inputs to the modeller.
6. Validate the chosen best model and determine the accuracy.

To align the given sequences, we have used multiple sequence alignment techniques for multiple templates selected, some of them include T-Coffee and Clustal-Omega. At some cases, we have chosen only one template, where we have used Needleman-Wunch algorithm to align the sequences and converted all of them into PIR format. The pipeline that we have implemented asks for sequence and automatically converts them into PIR format and saves query in .ali file and rest template sequences as .pir extension. The PDBs are automatically download once given the template IDs after script 1 execution. The program that we wrote has BioPython packages and uses NCBI pdb API call to download the PDB files given the template IDs. Alternatively, we can use REST API to download the pdb files. The REST API is of XML format.

For validation, we are using different tools available online and for visualization purpose we have used Chimera as our tool. The various online techniques that we have used to measure the accuracy of our model inclue TM-Score, Molprobity scores, and RMSD score.

All the steps are cleared explained clearly below where we have mentioned each of options that we have chosen for protein modelling.

The software distribution that we used for this project is described as below:

1. Test cases i.e. Protein queries are taken from CASP website. CASP10 and CASP 11 are chosen to select 10 proteins which include:
   * CASP11 targets T0856, T0843, T0806, T0837, T0792 and
   * CASP10 targets T0757, T0666, T0678, T0651, T0694
2. For template identification: BLAST, PDB, and SWISS-Model.
3. Sequence alignment: Needleman-Wunch algorithm, and T-Coffee
4. Software for protein modelling used is Modeller 9.17.
5. Various languages used are python 2.7 and JAVA (Needleman-Wunch algorithm)
6. Protein visualization software’s: Chimera, and Rasmol
7. Project Management and version control: Github

**Process automation:**

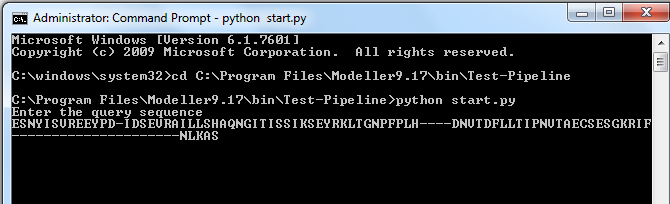
Implemented a sample pipeline where the process of using modeller is made lot easier compared to the standard approach. Instead changing inputs in the script files everytime, user needs to just enter the sequences initially and template IDs later for each of the script to run. The script automatically converts the given aligned query sequence into PIR format as saves it as .ali extension file. At each script execution, based on the output generated by the script and after evaluating the results, we given the template ID as input to the next script and the process is the same till the end. Sample screenshot for the pipeline is as below, where the program ask for user input query sequence.

Figure 1:start.py script; User entering query sequence

When the user enters the aligned sequence, the script process the sequence and ask for user to confirm by pressing enter, once the user clicks enter the first script of modeller starts running. Now for each script, once the execution is complete the console ask for user input. From script 2 users’ needs to enter just the template IDs and model ID for the last one accordingly after evaluating the log results generated by the modeller. The PDB files required for the template can be automatically downloaded when we use BioPython libraries available for python. So, taking these dependencies we can download the PDB files directly. Another option is we can use REST API calls provided by PDB website where are already available in XML format. So, we use them to download the PDB files required for running script2. Similarly, the pipeline process is continued for each of the proteins.

# Results

With the given ten CASP sequences, we divided the sequences among ourselves in the group and completed the process individually later we validated our results by exchanging the proteins we modelled among ourselves. For template search, we have used BLAST as our primary source and we played around with different option available with BLAST to match the best template that we can obtain from our resources. For most of the results that are obtained we have used PSI-BLAST option in BLAST and searched for template in PDB database. Each of the protein results are explained below separately and all the results obtained are discussed individually at first and then comparison is made with CASP results to show the accuracies of our protein model that we have generated.

## CASP 10 and CASP 11

For the process of modelling the proteins, we have chosen 5 query sequences for CASP 10: T0757, T0666, T0678, T0651, T0694 and CASP 11: T0856, T0843, T0806, T0837, T0792 respectivey we have followed the same procedure as discussed in the pipeline. Initially we have taken the query sequence and searched for template in BLAST with option as pdb database and PSI-BLAST searching techniques. Depending the results obtained and percentage of identity between the chosen templates and query sequence we have used either of pair wise sequence alignment with Needleman-Wunch algorithm and T-Coffee for multiple sequence alignment between query and templates which are explained clearly for each of the protein below with results obtained for each of the protein. For the purpose of pair-wise sequence alignment we have implemented a JAVA program from github and for multiple sequence alignment we have used T-Coffee and Clustal-Omega accordingly when required and for validation and refinement purpose.

### 3.1.1 Protein modelling using Modeller

We have taken 5 protein query sequences from CASP 10 i.e.

T0757, T0666, T0678, T0651, and T0694 respectively and from CASP 11 i.e. T0856, T0843, T0806, T0837, and T0792 as our query sequence individually and the results obtained are as follows:

* **T0757:**

we have taken the query sequence from CASP and used BLAST to obtain the template sequence, although we got couple of matching sequences but as result we took 4gak as out template and performed pair-wise sequence alignment between each of the query and template. Initially we tried with multiple sequence alignment but the TM-Score obtained for the protein sequence alignment is around 0.8, but when we tried pair-wise alignment with selected template we obtained TM-Score as 0.9629. the result of the modeller scores is as follows:

Filename molpdf DOPE score GA341 score

----------------------------------------------------------------------

Model1.pdb 1250.17883 -29520.33789 1.00000

**Model2.pdb 1209.11462 -29355.93359 1.00000**

**Model3.pdb 1170.82837 -29699.65625 1.00000**

Model4pdb 1374.11084 -29354.61719 1.00000

Model5.pdb 1231.56580 -29481.41602 1.00000

Model6.pdb 1167.89832 -29369.69531 1.00000

We have chosen model3 for our result which has better TM-Score compared to other models. We have also considered Model3 as a part of refinement which has better Molprobity score 2.02 compared to Model2 which has score of 2.43, so as a refinement we have reconsidered Model3 as our solution even though it has high DOPE score but least molpdf score.

* **T0666:**

For T0666 we have started the analysis and obtained multiple templates, after analysis we have chosen 3napA and 3ux4A as our resultant templates and performed multiple sequence alignment with query. The diagonalization matrix obtained as a comparison is as follows:

3napAA@23ux4AA@3

3napAA@2 264 2

3ux4AA@3 1 180

Based on the results which are bit confusing to choose the better template, we have performed modelling taking both the templates separately and after obtaining results we have chosen best template and model based on the TM-Score obtained for each of them respectively. Finally, we have chosen 3ux4A as the best matching template which and the results for model structures are as follows:

>> Summary of successfully produced models:

Filename molpdf DOPE score GA341 score

----------------------------------------------------------------------

Model1.pdb 1022.91730 -23121.25391 1.00000

**Model2.pdb 1053.24976 -23116.05078 1.00000**

Model3.pdb 950.99738 -23461.31836 0.99997

Model4.pdb 1070.12378 -23245.58594 0.99999

Model5.pdb 1003.10345 -23420.93164 0.99994

Based on scores, we have choosen model2 as our best model which as average of scores as well comparatively and obtained a TM-Score of 0.9042 and Molprobity score as 2.92.

* **T0678:**

we have performed pair-wise sequence alignment with the query protein taking 4epz as our template. The results obtained are better comparing to multiple sequence alignment that we have performed initially, later with refinement we have chosen pair-wise sequence alignment as best choice for this protein based on our template models. The results obtained for models are follows:

>> Summary of successfully produced models:

Filename molpdf DOPE score GA341 score

----------------------------------------------------------------------

Model1.pdb 643.56921 -18270.39844 1.00000

**Model2.pdb 613.73462 -18336.48047 1.00000**

Model3.pdb 1147.81177 -17304.24219 1.00000

Model4.pdb 595.51093 -18572.18359 1.00000

Model5.pdb 590.30353 -18337.23242 1.00000

We fixed model2 as our best model based on the results of TM-Score of 0.9222 and Molprobity score as 2.19.

* **T0651:**

We have performed multiple sequence alignment for this protein later we have analyzed our results after final decision of template and model selection, here we noticed multiple-sequence alignment performed better by few points compared to pair-wise sequence alignment. The results obtained for models of chosen template are as follows:

>> Summary of successfully produced models:

Filename molpdf DOPE score GA341 score

----------------------------------------------------------------------

Model1.pdb 1590.79980 -30183.49805 1.00000

Model2.pdb 1579.90295 -30485.20117 1.00000

Model3.pdb 1718.90918 -30320.98438 1.00000

Model4.pdb 1441.80750 -30527.74219 1.00000

Model5.pdb 1662.43384 -30278.29883 1.00000

**Model6.pdb 1435.20447 -30606.22852 1.00000**

We have finally decided to take model6 as our result model based on taking average scores of DOPE and molpdf. The TM-Score obtained is 0.97 and Molprobity score of 2.18.

* **T0694:**

Based on the results of BLAST, we have performed multiple sequence alignment of templates and query using T-Coffee and the results obtained for models are as follows:

>> Summary of successfully produced models:

Filename molpdf DOPE score GA341 score

----------------------------------------------------------------------

Model1.pdb 1435.19836 -38772.60547 1.00000

Model2.pdb 1422.11194 -38440.36719 1.00000

**Model3.pdb 1332.25159 -38742.54688 1.00000**

Model4.pdb 1465.64661 -38483.46484 1.00000

Model5.pdb 1380.60278 -38905.53125 1.00000

We have chosen Model3 as our resultant model taking average of scores obtained and we have achieved a TM-Score of 0.9576 and Molprobity score of 2.21 which is comparatively better solution based on analysis of CASP results which we will discuss later in the paper.

## Unnumbered list style

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

 (1)

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

**Fig. 1. Relation between τ and *t*.** This example has only two continuous Steppers, S1 and S2.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

**Table 1.**Benchmark results of the cascade oscillators model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |S| | Predicted cost | Timing | Predicted speed | Speed |
| 1 | S219.20(100%) | 68m43s | 1.00 | 1.00 |
| 2 | 29.10+219.10(~50%) | 35m13s | 2.00 | 1.95 |
| 4 | 219.20(100%) | 68m43s | 1.00 | 1.00 |
| 10 | 29.10+219.10(~50%) | 35m13s | 2.00 | 1.95 |
| 20 | 219.20(100%) | 68m43s | 1.00 | 9.5 |

This is table foot note sample text This is table foot note sample text This is table foot note sample text

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.The quick brown fox jumps over the lazy dog.

Acknowledgements

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Funding

This work has been supported by the …..

*Conflict of Interest:* none declared.

References

Alexandrescu,A. (2001) Modern C++ Design: Generic Programming and Design Patterens Applied. Addision Wesley Professional, Boston.

Dormand,J.R. and Prince,P.J. (1980) A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

Alexandrescu,A. (2001) *Modern C++ Design: Generic Programming and Design Patterens Applied.* Addision Wesley Professional, Boston.

Dormand,J.R. and Prince,P.J. (1980) A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

Alexandrescu,A. (2001) *Modern C++ Design: Generic Programming and Design Patterens Applied.* Addision Wesley Professional, Boston.

Dormand,J.R. and Prince,P.J. (1980) A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

Yoo,M.S. *et al*. (2003) Oxidative stress regulated genes in nigral dopaminergic neurnol cell: correlation with the known pathology in Parkinson’s disease. *Brain Res. Mol. Brain Res.*, **110**(Suppl. 1), 76–84.

Crenshaw, B.,III, and Jones, W.B.,Jr (2003) The future of clinical cancer management: one tumor, one chip. *Bioinformatics*, doi:10.1093/bioinformatics/btn000.

Auhtor,A.B. *et al.* (2000) Chapter title. In Smith, A.C. (ed.), *Book Title*, 2nd edn. Publisher, Location, Vol. 1, pp. ???–???.

Bardet, G. (1920) Sur un syndrome d’obesite infantile avec polydactylie et retinite pigmentaire (contribution a l’etude des formes cliniques de l’obesite hypophysaire). PhD Thesis, name of institution, Paris, France.