

Implementing Template-Based Structure Prediction And Refining with Rosetta

Project Description

The intent of this project is to implement a version of template-based protein structure prediction and analyze its performance. The implementation uses a combination of code I wrote from scratch, the pyrosetta API, and BLAST queries of the protein data bank. I initially planned to focus on template-based prediction, but the majority of my tests found a very good match for nearly the entire query sequence, so I instead tried to improve an initial template pass with rosetta refinement techniques, specifically the dihedral mover, small mover and shear mover. I tested the performance of several combinations of techniques on five proteins: 1AA9, 1IAQ, 2RH1, 2EVW, and 1B0H.

Why Is This Project Important?

As we have discussed in class, protein structure prediction is an important and already much-researched topic. It is not always possible to determine a protein structure experimentally. However, predicted protein structures can guide experiments toward an accurate structure. Predicted structures are also used in drug development and in predicting the function of a structurally undetermined protein. Existing techniques of protein structure prediction are roughly divided into *denovo* prediction and *template/homology-based* modeling. I researched phyre2 as an example of template-based modeling, and rosetta as an example of denovo prediction. My algorithm ultimately uses both strategies, a template-based first step which I implemented, and a second refinement step which uses pyrosetta.

Contributions to Solving the Problem

I began this project knowing that in this timespan I would not be able to implement a prediction algorithm as accurate as the current software packages built on decades of research and effort. However, it proved very useful both to my understanding of prediction methodology and pitfalls, and to my familiarity with rosetta. I am interested in continuing to experiment with protein structure prediction software, and hopefully finding improvements on current techniques.

Code Explanation

The majority of my work is in `utils.py` and `template_predict.py`. `utils.py` defines all the functions that implement template-based prediction, and `template_predict.py` runs a suite of

predictors. One of my first design decisions was to split the two files up in this way, so that `template_predict.py` would be able to easily run my template-prediction code, the `DihedralPredictor` from assignment 2, and new pyrosetta code I added (at the end of `template_predict`).

Some of my design decisions stemmed from difficulty with my first attempts at template-prediction. I tested my program on solved structures in the `pdb`, and most of my BLAST queries returned a good match for the entire query sequence, which not a good representation of using the program to predict a new, unsolved protein. Often when a protein structure is added to the `pdb`, other similar structures become more easily discovered and are added afterward. Therefore, to simulate solving each protein for the first time, I added an 'oldDB' option that runs the usual template prediction but filters out BLAST results that were added to the `pdb` after the predicted protein.

My main algorithm (`template_predict`) runs a BLAST query on the entire sequence of the predicted protein, and applies all alignments in the top result. It sets all phi, psi, omega, and optionally chi angles of the predicted protein to equal the angles of the corresponding residue in the protein returned by the BLAST query. All residues that are not matched as part of this alignment are added to a list and returned to the next phase. The next phase tests four possible secondary refinements: 1) run the `DihedralPredictor` only on the unmatched residues returned by the template phase, 2) run the `DihedralPredictor` with total backbone freedom, 3) run the `SmallMover`, 4) run the `ShearMover`. I modified the `DihedralPredictor` from assignment 2 to allow it to run on a restricted set of angles. The `SmallMover` and `ShearMover` are new rosetta code I added, similar to the `FragmentMover` in assignment 2.

The main difficulties I faced were with the lack of documentation for most of the software and APIs I used. PyRosetta only has python documentation for a few methods. The rest are only defined in the C++ documentation for the entire rosetta code base, and the python types and usage must be inferred from C++. Additionally, the format returned by the BLAST queries was inconsistent and the alignment was often wrong. Specifically, the "hit sequence" would be a correct fragment of the source protein, but the "hit-start" would be off by ~6-7 indices from the correct residue number. More rarely, the hit-sequence itself was inconsistent with the sequence in the `pdb`. I added code to check whether the alignment information in each hit was accurate, attempt to fix it, and discard that result if fixing was impossible.

Results

After running a combination of predictor on each flattened test protein, I aligned and compared the result to the correct structure in the protein data bank, and recorded the all atom root mean square distance. My final results are listed in `collate_results.txt` as "[residue] | [predictor

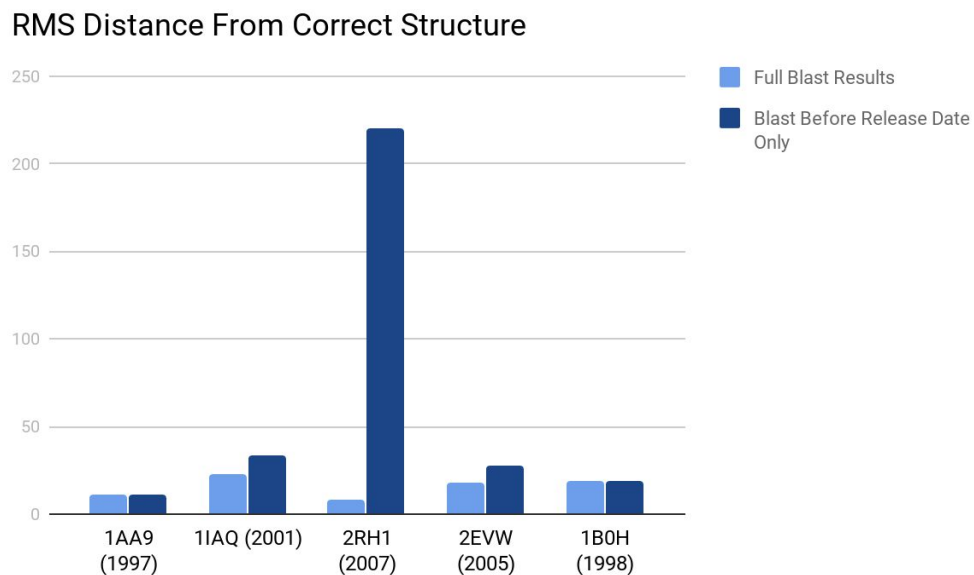
methods] | [rmsd of predicted result and correct structure]", and summarized below. These results can be reproduced by running "python template_predict.py". There are no arguments, but you must install pyrosetta as well as a couple other python modules through pip install. I find it helpful to run 'python template_predict.py | grep -E "1AA9 |1IAQ |2rh1 |2EVW |1B0H "' to clearly see all my result output lines without the clutter of all the rosetta output.

RMSD from Correct Structure



	1AA9	1IAQ	2RH1	2EVW	1B0H
Full blast	10.727	22.771	7.827	17.791	18.559
Full + chi	10.571	22.811	7.435	17.822	18.467
Full + dihedral restricted	10.743	26.480	7.835	45.921	18.581
Full + dihedral full	28.936	51.967	67.427	54.523	50.273
Full + small mover	10.853	21.252	10.985	17.291	19.486
Full + shear mover	10.658	22.853	7.451	17.680	18.544
Old Blast	10.727	33.248	219.879	27.807	18.559
Old + chi	10.572	33.333	219.914	27.745	18.467
Old + dihedral restricted	10.744	35.356	143.911	29.687	18.581
Old + dihedral full	34.203	54.403	174.566	40.946	71.970
Old + small mover	10.206	30.643	223.780	26.056	19.695
Old + shear mover	10.516	33.011	220.082	28.186	18.758

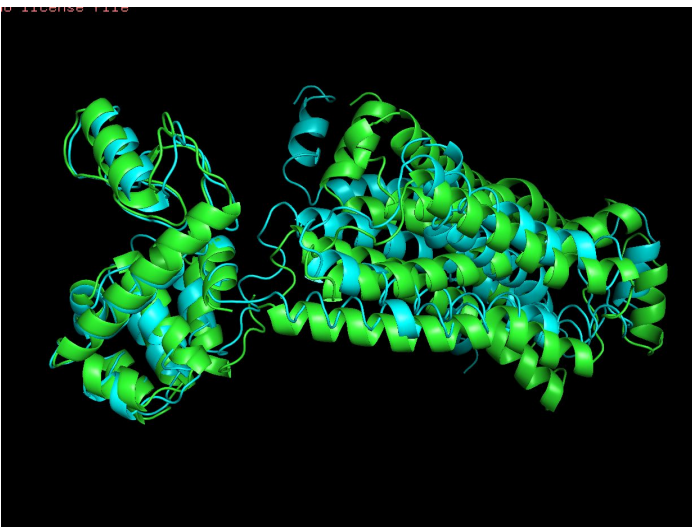
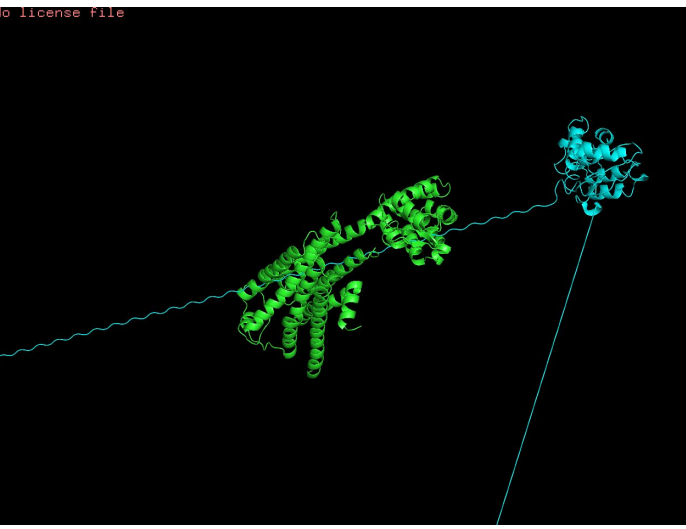
One consistent finding was that applying the chi angles of the template residue had little to no effect on the accuracy of the result. This is not that surprising since the backbone is far more important to the overall structure, but it was fairly straightforward to implement and worthwhile to test. Another consistent finding was that none of the small refinements improved upon the template result. However, with more time, I believe that by adjusting the parameters and number of iterations of these predictors, I could make them improve the structure.

The most interesting findings concerned the difference between using all BLAST results or only BLAST results from before the release date of the predicted protein. As the chart below shows, there was no difference for two of the proteins, about a 50% drop in accuracy for two of the proteins, and an enormous drop in accuracy for 2RH1. Interestingly, the proteins that were more difficult to predict with old results do not correlate with their own release dates.



The reason for the failure of the old BLAST result predictor for 2RH1 is clearly evident from the final predicted structure. It is shown below with 1IAQ, which became slightly less accurate when only using old results, but still worked fairly well.

	
<p>1IAQ correct structure (green), 1IAQ prediction with full BLAST results (blue)</p>	<p>1IAQ correct structure (green), 1IAQ prediction with old BLAST results (blue)</p>

	
<p>2RH1 correct structure (green), 2RH1 prediction with full BLAST results (blue)</p>	<p>2RH1 correct structure (green), 2RH1 prediction with old BLAST results (blue)</p>

I have not included images of all the results, but they are similar to 1IAQ, in that BLAST found a decent match for the entire sequence both with old results and all results. Moreover, 2RH1 was one of the last proteins I tested. Because of this, I decided to focus on small refinements after one template match instead of making my template step more sophisticated. To accurately predict structures like 2RH1, it should divide the predicted protein into smaller fragments and look for multiple matching templates. 2RH1 with old BLAST results returned a match for only a small portion of the sequence, and small refinements were not enough to fold the rest of the sequence.

Improvements

There are many improvements I would like to make and other tests I would like to run. First and foremost, I will improve the template step. My initial algorithm would be

```
// while there are unmatched portions
    // BLAST search longest unmatched fragment (initially entire sequence)
    // apply first match
    // add new unmatched fragments to list
```

I would also like to introduce fragment predictors to the post-template step. Ultimately, my algorithm would be as it is defined above, but once the list only contained fragments of length 9 or smaller, I would switch to rosetta fragment prediction, restricted to these locations.

Finally, I would like to experiment with running the current refinement steps (dihedral, small, shear) for different iterations and with tweaked parameters. Overall they did not improve upon the template result, but I only ran them for 1000 iterations, which is perhaps not enough to get accurate Monte Carlo results. It makes sense that if a randomized predicted is run, for few iterations, on a structure which is already very accurate, the result could get worse before it gets better.

I learned a lot from this project, especially about rosetta and BLAST queries. Finding the best match for sequence is a much harder problem than I initially realized. I feel like I made a breakthrough at the very end of my implementation attempts, and I have many ideas for how to improve my algorithm. I would be very interested to continue working on protein structure prediction.