Karl Gutwin
May 15, 2005
18.336

A Parallel Implementation of a Higher-order Self Consistent Mean Field

Effectively solving the protein repacking problem is a key step to successful protein design. Put simply, when designing a protein, one starts with the intended backbone structure, on which a sequence of residues is to be placed with the ultimate goal that that sequence natively folds to the desired structure. Important features of these structures include that the chosen sequence packs in such a way that a minimum of unfilled space is left within the core of the protein, as well as there are no significant spatial clashes between atoms. An early method for solving this problem, and still widely employed, is the rotamer packing method.

A *rotamer* is the name given to the collection of bond angles which uniquely define a particular conformation of a residue. Rotamers are usually found in sets called *libraries*, where members of these sets are used, computationally, to discretize the problem of searching over infinite conformational space over many dimensions. Rotamer libraries are often generated through statistical analysis on known molecular structures, and can describe a significant fraction of the accessible conformational space of a given residue.

The rotamer packing method, at its core, needs to choose the best rotamer out of a library for each position in a protein chain. Here, the best rotamer is defined as the one which, out of all other possible rotamers, produces the least amount of clashes with other residues and stabilizes best the overall structure. Unfortunately, as may be apparent, this problem is hard to solve simultaneously, especially in a moderately sized protein of 250 residues (and approximately ten rotamers per residue). Therefore, several algorithms are available which reduce the runtime significantly. One of these algorithms, on which I will focus the remainder of this paper, is called the Self-Consistent Mean Field.

The first paper describing this method was published by Koehl and Delarue in 1994.[1] In it, they describe their application of mean field theory. To be succinct (a detailed explanation of this theory is given elsewhere[2]) the theory states that the effective potential of a given rotamer is given by the sum over all other rotamers of the probability that that rotamer exists in the final structure multiplied by its true pairwise potential. This probability can be subsequently calculated using the Boltzmann distribution. These relationships form the basis for an iterative process whereby effective potentials are calculated for each residue and then probabilities (stored in what the authors term the *conformational matrix*) are calculated on the basis of the effective potentials. This process converges over 20 or fewer rounds. At the end of the process the best rotamer can be selected simply by choosing the one with the highest probability in each residue position.

The advantage of this procedure is that it operates in linear time with respect to the number of residues. Unfortunately, it works under an assumption that residues which exhibit statistical dependence on one another will eventually choose the right rotamer through this iterative process. However, this assumption has not been shown to be true. To put forth some evidence in resolving this, and perhaps to create a more accurate version of this method, I added explicit dependence to the self-consistent mean field (SCMF).

In the original SCMF definition, each group of rotamers was a single residue. In my new definition, each group is actually comprised of the rotamers from several

1 Koehl, P and M. Delarue. *J Mol Bio* **239**, 249-275 (1994).
2 Koehl, P and M. Delarue. *Curr Opin Struct Biol* **6**, 222-226 (1996).

residues. Therefore, the number of rotamers is now the product of the number of rotamers from each comprising residue, and each new *rotamer set* contains a rotamer from each comprising residue. Otherwise, everything is computed identically. This has the benefit that when the number of rotamers in each *residue group* is set equal to one, the algorithm computes the original SCMF. This will be useful in final comparisons.

D	***Groups***	***Sets/group***	***Total sets***
1	100	10	1000
2	50	100	5000
3	34	1000	34000
4	25	10000	250000

Table 1. Exponential growth of higher-order SCMF. The example given is with a 100 residue protein with an average of 10 rotamers per residue.

One major issue with this method is that as the size of each group grows (what I term the *dimensionality* of the problem) the number of rotamer sets grows exponentially. Table 1 shows this growth for a small protein. As one of the steps to mitigate the computational impact of this growth, I have implemented this solution using parallel computing. My implementation uses straight C/MPI with a minimum of outside libraries (only libm).

In preparing the parallel implementation of this, some hurdles had to be overcome. The first involved memory issues. The original SCMF precalculated all the per-residue true potentials, since these did not change over the course of the calculation. However, it became quickly apparent to me that the size of this calculation would exceed the memory capacity of our cluster if the dimensionality went much higher than two or three. To mitigate this, I elected to recalculate all true potentials at each iteration. This had the advantage of allowing larger calculations to continue, but at the cost of a significantly longer runtime. The second hurdle involved load balancing. The basic unit of the computation is the rotamer set. However, different residue groups have different numbers of rotamer sets – sometimes very significant differences as the dimensionality grows larger. To mitigate this, I had to divide the residue groups among the processors such that the number of sets per processor was roughly equal. This did not help when the dimensionality grew large, however, since some groups would have more sets than could be distributed among the other processors. This created a bottleneck where adding more processors did not decrease runtime.

D	***RMSD***
1	18398.3
2	18392.3
3	18323.7
4	18318.4

Table 2. RMSD comparisons between template structure and best SCMF calculated structure. Tests done using PDB ID 1NLM.

In order to evaluate the results of my code, a number of different metrics could be used. Unfortunately I only had sufficient time to test, in a somewhat limited fashion, a simple but effective metric. RMSD (root mean square deviation) is a common metric in the protein structure field since it quickly provides a number which can be easily compared between other structures of the same molecule Table 2 reports the results from calculating the RMSD between the template structure and the best SCMF calculated structure with various dimensionalities. It is clear that the RMSD does decrease as the dimensionality increases; however, it is interesting to note that the RMSD between various recalculated structures is much higher than the difference between their RMSDs against the template structure. This suggests that increasing the dimensionality does not uniformly increase the number of "best" choices – in other words, the changes from D=2 to D=3 are not simply a few better choices but a collection of different choices which overall total a better score.

Any report on such a parallel process would be incomplete without some mention of runtime. It is interesting to note that I anticipated nearly a p-fold speedup given the nature of the calculation which is quite simply parallelized. However, the bottlenecking issue did come into play quite strongly, especially at the higher dimensionalities. Figure 1 shows these results. At D=1, the curve is nearly linear, reflecting p-fold speedup. However, at higher dimensionalities, the curves become bent, suggesting some alternate effects. The bottlenecking effect is unpredictable and appears to have been shown, in this instance, at NP=8 for D=3 as well as NP=16 for D=2. One test run was done at D=4, for NP=16 only, which took 7820 seconds. This test was not repeated at lower processor numbers, and no tests were performed at higher dimensionalities, for reasons of time constraints.

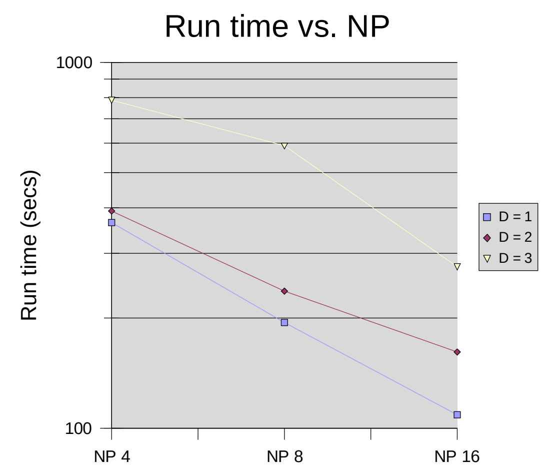

Figure 1. Runtime analysis performed on PDB ID 1NLM.

There are still many aspects of this project which are left open for improvement. One clear item is that during the course of this algorithm, many rotamer sets are generated which ultimately have such a low conformational matrix value that they never could be viable final candidates. To cull these sets would dramatically decrease processing and would allow for easily doubling the maximum possible dimensionality. The program is structured such that removing these sets would not be too difficult; however, no time was available to implement such a step. Also, the bottlenecking issue could be resolved by breaking up residue groups; however, culling those sets which are very large would probably have a much greater impact. Finally, it should be mentioned that the current version of the program creates groups by sequential addition; this is not the best strategy structurally speaking since residues often interact over large sequential distances. A clustering algorithm could identify those clusters of residues which would benefit most from being grouped together.

Overall, this project has shown success in implementing a parallel, higher-order self-consistent mean field. Preliminary results do suggest that such an implementation produces better results overall than a similar, first-order method. Further improvements are possible which could allow for a further increase in dimensionality as well as more accurate clustering. While it is unlikely that this method will ultimately gain widespread use (due to other, advanced algorithms which perform well on single-processor workstations) this may prove to be the most accurate method available.