Protein Folding Challenge and Theoretical Computer Science

Somenath Biswas
Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur.
(Extended Abstract)

September 2011

Almost all functions of a living organism at its cellular level are carried out through various classes of proteins. Our body consists of many kinds of *cells*, and each cell carries a copy of the *genome*– which is the program which a cell executes for its functioning. The genome is a polymer, or a chain; for us, it is a sequence of four kinds of molecules, known as *nucleotides*, and these four kinds are denoted as A, C, T, and G. Thus, a genome or DNA is a string over the alphabet $\{A, C, T, G\}$ (the human genome has about 3×10^9 symbols). Certain subsequences of the genome sequence are *genes*. A gene is *translated* into a *protein*. Each triplet of nucleotides is a code of one of twenty amino acids. For example, TTT TAC TGC GGC is translated as the sequence of four amino acids: Phe Tyr Cys Gly. Thus, from a gene, the cell makes a sequence of amino acids; such a sequence of amino acids is a *protein*.

Each protein, which forms as a chain, *folds up into a unique shape,* unique for that sequence of amino acids. By *structure of a protein*, we mean this unique 3D shape of the protein. The functionality of a protein is due to its 3D shape, that is, its structure. By knowing the structure of a protein, we understand its function.

It is relatively easy to find out the chain of amino acids that defines a protein. The protein folding problem is: *given a sequence of amino acids, determine the 3D shape that this sequence will give rise to.* What we are asking for is an algorithm, and as computer scientists we want an efficient, that is, a polynomial time algorithm.

In a living cell, many many chains of amino acids are getting formed all the time. Such a chain folds into its unique 3D shape usually within a few miliseconds, and then it functions as it should. Thus, nature seems to use an efficient algorithm to carry out protein folding.

The atoms in a protein molecule attract each other. As they come closer, there will be torsions and compressions of the chemical bonds which will limit this coming closer. Given a configuration, that is, a positioning of various atoms in the 3D space, we can compute the potential energy of the configuration. Any natural system tends to go towards its min-

imum energy configuration. Protein folds as it goes to its minimum energy configuration. Therefore, to find folding, we need to find out that configuration amongst all possible configurations, which has the minimum energy. Thus, the *protein folding problem is essentially an optimization problem.*

The general computational problem is:
Given a set of atoms, and their connectivity, and a potential function, find that configuration which has the globally minimum energy. It has been shown that this problem is NP-hard. Therefore, it is extremely unlikely that there is an efficient algorithm to solve this problem. And yet, *nature seems to have an efficient algorithm!*

It has been observed beyond any doubt that all proteins fold very rapidly, that is, nature obtains the minimum energy configuration very efficiently. When first studied, biochemists found this to be paradoxical, because there is an astronomically huge number of configurations, and taking such configurations to be metastable, they computed the time required to come to the folded configuration. This computed time was found to be many orders of magnitude more than the observed time for protein folding. This paradoxical situation is called the *Levinthal's paradox.*

Just because the search space is huge, say, exponential, a search problem in that space need not be hard. There are, in general, exponentially many paths in a graph between s and t, and yet we have polynomial time shortest path algorithms. A restatement, however, does appear paradoxical.

Seemingly, nature has an efficient algorithm for protein folding. How costly is it if we simulate it? (Here, we follow the argument given in [NMK94]. There are two issues. Let N be the instance size:

- In simulation of a system's dynamics on a computer, we use discrete steps. Let $F(N)$ be the frequency at which system states (that is, velocities and accelarations) are recalculated.
- Let $S(N)$ be the number of steps used by the simulating algorithm to compute one single discrete time step.

If nature's algorithm takes $T(N)$ units of time on the instance, then the simulation will require $F(N)S(N)T(N)$ steps. Thus, if $F(N)$ and $S(N)$ are polynomials in N, then the simulation algorithm will work in polynomial time. For protein folding, it has been shown $F(N)$ is actually a constant, though of a small value, about 10^{-15}. And $S(N)$ is $O(N^2)$, thus both $F(N)$ and $S(N)$ are bounded by a polynomial in N. We therefore conclude that if nature has a polynomial time algorithm, then the simulation also works in polynomial time. On the other hand, if $\mathbf{P} \neq \mathbf{NP}$, as the protein folding problem is shown to be NP hard, nature does not have an efficient algorithm to carry out protein folding. How then do we account for the observed rapid folding of natural proteins?

A plausible resolution of the paradox can be:

1. Nature uses an algorithm which is NOT a polynomial time algorithm.
2. An algorithm which is exponential time in the worst case, may be efficient for a subset of the input domain.
3. Natural proteins come about precisely from those amino acid sequences for which, when they are given as inputs, the algorithm employed by nature will work efficiently.

If we are able to say eficiently, given a sequence of amino acids, whether or not it will fold rapidly into a unique 3D structure, and if it does, what is that unique 3D structure, it will be of immense value. If we find out nature's algorithm, then we are done. However, till today no one has any clue about nature's algorithm. We in CS can take a *guess and check* approach: Try out your favourite search algorithm and see if works efficiently on natural proteins.

It is more likely than not that nature's algorithm is a randomized algorithm. Some biochemists have argued pursuasively that the algorithm used is what we call the Metropolis algorithm [SK91].

We recall that in a general combinatorial optimization problem, every instance I of the problem has an associated, usually implicitly, search space, say S_I. We also have a function f which maps S_I to numbers. Given any element $a \in S_I$, we can efficiently compute the value of $f(a)$. The optimization problem is to find that element p of S_I, at which the function f is minimized.

For the protein folding problem, I is a given chain of amino acids, S_I is the set of configurations of the chain, and for a configuration x, $f(x)$ is the energy of the configuration.

The Metropolis algorithm [MRR53] is a randomized search heuristic which can be seen as a biased random walk in the search space, the bias being such that the algorithm will generally favour going from a search point to a search point with less cost; however, it can also go to a neighbouring point where the cost increases, but such transitions will happen with less probability. Such transitions ensure that the search will not get trapped in local minima.

The key question, therefore, is: will the Metropolis algorithm find efficiently minimum energy configurations efficiently, i.e., in time polynomial in the size of the amino acid chain? Answering this question may need going deep into certain biochemistry issues: instead, what we can certainly do is to characterize the set of instances on which the Metropolis algorithm is *guaranteed* to work efficiently, and then leave it to the biochemists to figure out if the instances generated by protein sequences form a subset of this set or not.

What the Metropolis algorithm does on an input instance is to run a Markov chain. At the tth step, the chain can be in any of the configurations with a certain probability; in other

words, there is a probability distribution σ_t on the space of configurations, the chain at time t will be in a configuration as per the distribution σ_t. Our question can then be framed as: after fixing a polynomial p, which are the instances I such that at a time $t \leq p(|I|)$, the minimum energy configuration of I has a large probabilty as per σ_t? If we term the minimum energy configurations as goal states, the question in Markov chain phraseology will be to characterize the set of instances for each of which the Markov chain defined by the Metropolis algorithm has an expected first passage time to the goal state bounded by a polynomial in the instance size.

One conjecture came from the biochemists Sali, Shakhnovich and Karplus [SSK94a], [SSK94b]. They had used the Metropolis algorithm to search for the minimum energy conformations of chains of beads in the lattice model of protein folding. Based on their computational experiments, they concluded that the Metropolis algorithm would find the minimum energy conformation of a chain of beads within an acceptable time scale if and only if there is a large gap between the energies of the minimum energy conformation and that of the second minimum. Clote [Cl99] attempted to support this conclusion by a proof that the mixing time of the underlying Markov chain would decrease as the gap in the energies of the minimum energy conformation and that of the second minimum increased. He was able to show that an *upper bound* on the mixing time does indeed decrease as the energy gap increases. We state a recent result [NB10] which shows that the mixing time *itself*, however, is a nondecreasing function of the value of the energy gap. Therefore, our result contradicts what Clote had attempted to prove. It is clear that we need more work in this direction.

However, there has been some progress at another level. A fundamental result in the the theory of Markov chains is that if we have a family of *ergodic* Markov chains, then for each chain in the family, there is a stationary distribution such that if we run the chain for a sufficiently long time, starting from any initial distribution, the chain will come close to its stationary distribution. Therefore, a sufficient condition for the Metropolis algorithm to be successful for the protein folding problem is: each chain in the family of chains defined by the algorithm comes very close to its stationary distribution in time polynomial on instances of interest (in other words, the family is rapidly mixing) and the probability of the goal state is high in the stationary distribution. However, for the protein folding problem, what we require is that the expected first passage time is small for every chain on instances of interest. It has been shown [SRB10] that this *necessary condition* is actually equivalent to the sufficient condition mentioned above for families of chains defined by the Metropolis algorithm. Rapid mixing of Markov chains has been an area of active research in recent years; several powerful techniques like coupling, canonical paths, etc. are available for arguing about rapid mixing. The result we have cited above makes all these techniques now available for studying the performance of the Metropolis algorithm for combinatorial optimization problems, in particular, for the protein folding problem.

References

[Cl99] Peter Clote, Protein folding, the Levinthal paradox and rapidly mixing Markov Chains, *Proceedings of 26th International Colloqium on Automata, Languages and Programming*, LNCS 1644 pp 240–249, 1999.

[NMK94] J Thomas Ngo, Joe Marks, and Martin Karplus, Computational Complexity, Protein Structure Prediction, and Levinthal Paradox, in *Protein Folding Problem and Tertiary Structure Prediction*, Kenneth Merz, Jr. and Scott Legrand (eds.) 1994.

[MRR53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, Equations of state calculations by fast computing machines *Jl. of Chemical Physics,* 21:1087–1092, 1953.

[NB10] Apurv Nakade and Somenath Biswas, Effect of increasing the energy gap between the two lowest energy states on the mixing time of the Metropolis algorithm, 2010. (Communicated)

[SSK94a] Andrej Šali,Eugene Shakhnovich and Martin Karplus, How does a protein fold? *Nature,* Vol. 369, pp 249–252, 1994.

[SSK94b] Andrej Šali,Eugene Shakhnovich and Martin Karplus, Kinetics of protein folding: A lattice model study of the requirement for folding to the native state, *Jl. Molecular Biology*, Vol. 235, pp 1614–1636, 1994.

[SK91] J. Skolnick and A. Kolinski, Dynamic Monte Carlo simulation of a new lattice model of globular protein folding, structure and dynamics, *Jl. Molecular Biology*, Vol. 221, pp 499–531, 1991.

[SRB10] Swagato Sanyal, Raja S. and Somenath Biswas, Necessary and Sufficient Conditions for Success of the Metropolis Algorithm for Optimization *Proc.of 12th ACM GECCO Conf.,* 1417–1424, 2010.