guha@stanford.edu
December 9, 2002

Constructing Nature's Building Blocks:

An Analysis of Computational Protein Design Today

by Guha Jayachandran

CONTENTS

INTRODUCTION

"*We cannot command nature except by obeying her.*"
-Sir Francis Bacon (1561-1626)

Proteins are the building blocks of life, "nanomachines" that do everything from comprise our muscle to serve as enzymes. Computational protein design attempts to, given a three dimensional protein structure, derive an amino acid sequence coding a protein that will adopt that structure. It is in a sense the opposite of the protein folding problem: while protein folding studies aim to go from an amino acid sequence to a native three dimensional structure, protein design aims to go from a structure to sequences.

Evolution, over the course of millennia, has "designed" and refined countless proteins. Mutations in the genetic code, which codes for proteins, allowed for new proteins to arise. If a protein bestowed evolutionary advantage on an organism, it was likely to be passed on. Evolution just happened, though—why would we want to design proteins computationally, or design them deliberately at all?

The answer to the first question is that the most successful experimental methods, involving directed protein evolution in vitro, are constrained to searching spaces of at most 10^6 sequences [17]. Computational design can be used to trim the sequence space to areas of high interest, and then its result sequences can be verified experimentally.

Now to the second question of why to design proteins at all. Among the motivations for protein design are understanding the relationship between sequence and structure, aiding structural and functional genomics, and creating proteins with desired functions.

Protein design furthers our understanding of the relationship between amino acid sequence and three-dimensional structure in a very fundamental manner. Since protein design, as we will see, typically involves mutating sequences with known structure, it allows us to learn how particular changes in a sequence affect the structure, and thus, more generally, how sequence leads to structure. Also, there are thought to be thousands of times more genes than three dimensional protein structures—protein design can shed some light on this degeneracy as it tries to find all the sequences that could form a single structure. Protein design is, at its most

fundamental, an attempt to make a sequence for a structure—so fully solving the relationship between structure and sequence would imply solving the protein design problem.

Structural and functional genomics have been active areas in the post-Human Genome Project era. Protein design contributes to these fields by increasing the variety and number of proteins available for their techniques. In particular, with a variety of proteins produced by protein design, fold recognition techniques have a greater reference set to base their recognition on. Multiple sequence alignments, used to identify functionally important regions by looking for commonalties in the sequences of proteins, will as well be aided by the increased number of sequences available for a structure due to protein design.

Protein design indeed offers great promise and has come a long way, but it is yet an infant field. For the above aims to be achieved, much work must be done and obstacles overcome. The methods in use today must see improvements or must be replaced, as otherwise their drawbacks will keep design mired in oversimplification and a capacity only for small proteins. In this paper, we will see what has been accomplished thus far in protein design, how it is done, what drawbacks to current approaches are, and what avenues of improvement I have thought up.

HISTORY AND STATE OF THE ART

Here, I will summarize not the progression of the method of protein design, but a timeline of major accomplishments, for the purpose of putting the challenges and critiques of later sections in context. Protein design has come a long way since the birth of its current form in the early 1980s. As DeGrado wrote in 1997, "Not long ago, it seemed inconceivable that proteins could be designed from scratch. Because each protein sequence has an astronomical number of potential conformations, it appeared that only an experimentalist with the evolutionary life span of Mother Nature could design a sequence capable of folding into a single, well-defined three-dimensional structure" [4] (more on the "astronomical number" of conformations in Approach).

In his 1981 landmark paper envisioning the field of nanotechnology (which he referred to as "microtechnology"), Drexler started his abstract with, "Development of the ability to design protein molecules will open a path to the fabrication of devices to complex atomic specifications" [6]. He portrayed protein design as a problem for "engineers" that could be solved more easily than the "scientists'" problem of protein structure prediction, as designed sequences could be very heavily biased towards stability in their intended structure.

Two years later, Pabo defined the inverse folding problem, and the notion was introduced of screening sequences for those that will adopt a specific structure [12]. A solution to the inverse protein folding problem would give the full understanding of how protein design could be done, and as things stand, protein design offers insights on the inverse protein folding problem [9].

In 1987, a four helix bundle protein was engineered [7,16]. Also during the 1980s, it was confirmed that even a single amino acid modification could raise the stability of a protein beyond that evolved through natural selection, though a number of mechanisms. The four helix bundle of [7], for instance, had a stability of 22 kcal/mol, far greater than the 9 kcal/mol that is usually considered the high end for proteins of that size [13,11].

The stage was set for a computational breakthrough. Even with Moore's law at the backs of computational protein designers, the demands of protein design calculations were still too immense (and still are for many tasks, as we will see). In 1987, Ponder and Richards introduced

and justified the fixed backbone and discretized rotamer libraries assumptions to reduce computational requirements. These will be described in detail in Approach.

Most recent work has made use of such assumptions and of the general approach described in the next section. Major accomplishments it has seen to date include the first fully automated design of a novel sequence for a whole protein in 1997 by Dahiyat and Mayo [3]. They screened (included in their ORBIT algorithm) a combinatorial library of 1.9×10^{27} amino acid sequences for compatibility with the target structure, a beta-beta-alpha protein motif based on the polypeptide backbone structure of a zinc finger domain. The designed sequence had low identity to any known protein, and when experimentally translated, the resultant structure was found to agree well with the target. Localized core, surface, and interface design have seen great success. Just earlier this year, de novo design of a biocatalyst was achieved [2]. In later sections, I will mention specific techniques used in these achievements. Despite was has been accomplished, though, there are as we will see, significant limitations to current methods, and a pressing need for improvement.

OVERVIEW OF APPROACH AND ITS RATIONALE

Accomplishments in computational protein design, such as those of the Mayo group described above, have followed a fairly standard approach [14]. Here, that approach is outlined with an explanation for why it developed the way it did. In the second half of the section, I give a concrete example of the process.

There are an astronomical number of sequences and structures, with all the various degrees of freedom that these have [14]. Choosing from these possibilities an optimal (even if not the optimal) sequence solution is the challenge. This is why the fixed backbone and rotamer library approximations were introduced by Ponder and Richards. They posited that the problem would be made much more tractable by assuming that the backbone could not move at all and that side chains could only take certain allowed conformations (listed in a “rotamer library”). With these approximations, the search space becomes discrete and computational protein design becomes a problem of searching for the lowest energy combination of side chain rotamers for the fixed backbone.

Even with these major simplifying assumptions though, the sequence structure space remains incredibly huge. As Pokala and Handel point out, a100 amino acid protein would have—with just two rotatable bonds and five conformations at each position—$(20\text{a.a.} \times 5^2)^{100} = 500^{100}$ sequence structure solutions. Even here, and especially for larger proteins, an exhaustive search for structure-compatible sequences is not possible. Various search algorithms and simplified energy functions (for comparing two structures) have been put to work on finding solutions from the large search space. These search methods and energy functions will be analyzed in later sections.

A concrete example may prove helpful here. I will use the SPA (sequence prediction algorithm) algorithm [15] as used in Larson’s Genome@home studies to provide examples of specifics [10]. Though most computational work has used the fixed backbone and rotamer library approximations, recently some, including Larson, have somewhat relaxed the fixed backbone assumption. This just adds one step to the start of the process. The procedure outlined below includes this additional preliminary step for completeness, and the rest is fairly representative of other work.

First, a protein with structure similar to that desired (or with the structure desired, but with less specificity or some other chemical property different) is chosen. Based on its crystal structure, its dihedral angles are perturbed randomly up to five degrees. A simple Monte Carlo technique is used to push the angles back towards the crystal until it is within the target root mean square deviation (RMSD) from the native structure (Larson found that a RMSD of 0.2 Angstroms ultimately produces as much final sequence diversity as a threshold of 2 Å, but in [9], he uses a threshold of 1 Å). There are now 100 backbones derived from the original.

The rotamer library has 3,008 rotamers and was constructed by examination of hundreds of thousands of crystal structures. It is divided into 200 sets, each consisting of, on average, 150 rotamers allowed for a certain type of position (such as at first carbon). SPA builds 300 initial models from a single backbone using the rotamer library, essentially starting off at a random point in sequence space. The models' energies are calculated (a combination of the Amber potential function [18] with OPLS nonbonded parameters [8] is used) and ranked. Since only the relative energies are needed for ranking, □□G values (change between each other in change in Gibbs free energy) are what really are of import.

Next, a genetic algorithm is used to recombine between the lowest energy models, in order to try making ones of greater fitness, with greater fitness in this case defined by even lower energy. The algorithm allows for swaps of single residues between models. 300 recombined models are created, and their energies ranked. Then these go back into the genetic algorithm. This genetic algorithm outer loop, surrounding the inner loop of 300 energy calculations, is done 200 times, for a total of 60,000 energy calculations in the genetic algorithm step. The entire cycle—from initial 300 models comprising a random start in sequence space to completion of 200 rounds of genetic recombination—is done 30 times.

That 30 iteration process is done for each of the 100 backbones initially made by perturbing the original backbone. The final result sequence set for the target structure can include designed sequences for all 100 backbone variants.

ANALYSIS OF SEARCH ALGORITHMS

A genetic algorithm is, of course, not the only method of searching sequence structure space. The various search algorithms employed each have advantages and disadvantages. The major ones are Monte Carlo algorithms, genetic algorithms, Self-Consistent Mean Field, and Dead End Elimination [14,5].

Monte Carlo algorithms

This is the simplest technique. Under a Monte Carlo method, a structure is randomly changed in residue type or rotamer at a position. If the change is energetically favorable, it is accepted. If it increases energy, then it is accepted or rejected based on the Metropolis criterion, basically a weighted a coin toss where the weights are calculated using the Boltzmann relationship between probability and energy difference for the given temperature. By sometimes allowing energetically unfavorable moves to be made, the search is able to escape local minima. Simulated annealing is a nice modification to the Monte Carlo method, whereby the temperature is gradually lessened throughout the search to lessen the probability that energetically unfavorable moves will be accepted late in the search. Two advantages of using Monte Carlo are that it is simple and can deal well with increases in the dimensionality of the search space, since

it is not doing anything exhaustive. A key disadvantage that it does not necessarily converge to a global minimum.

Genetic algorithms

Genetic algorithms comprise a second type of stochastic method. Under it, structures are mated, swapping sequences and rotamers to make hybrids that often have lower energy. Mutations that alter a residue or rotamer conformation can be introduced randomly to increase sequence diversity. Low energy hybrids are used to repeat the process, until convergence is observed. Genetic algorithms, as stochastic algorithms, share the disadvantage of Monte Carlo that they do not guarantee convergence to a global minimum. They are especially good though at exploring large sequence spaces and, in practice, are actually strong at overcoming local minima. Desjarlais and Clarke note that their effectiveness arises from the fact that the recombination between structures can bring together two spatially separate segments that were each optimized in separate structures, so that the overall structure with them both is more favorable than with only one or the other [5].

Self Consistent Mean Field

This method, unlike the stochastic ones examined thus far, is deterministic. Whereas Monte Carlo and genetic algorithms can give different results depending on what start points they are given, Self Consistent Mean Field optimization will always converge to the same solution. Limitations of space prevent me from giving too much detail on the method, but its sketch is as follows. A template is set up with many rotamers at easy residue position of the template. Each rotamer is given a uniform Boltzmann probability, and the rotamer probabilities are used to get a weighted average energy for each rotamer and each position. A new Boltzmann probability is calculated for each rotamer using the energies, and the process is repeated until the probabilities converge. Though this method will always give the same result, it too is not guaranteed to produce a global minimum.

Dead end elimination

This is a deterministic search method that, as a quasi-exhaustive technique, does guarantee convergence to a global minimum. It has been used with great success, including in the design of a biocatalyst described earlier. It uses the elimination of high energy rotamers or rotamer combinations from consideration to prune the search space. A few variant criterions have been developed for it. The original one was that if the lowest energy structure using a particular rotamer was higher than the highest energy structure using a different rotamer, then the first rotamer could be eliminated. Goldstein introduced the more liberally pruning criterion that if the energy of a structure containing one rotamer is always lowered by replacing that rotamer with another rotamer, then the first rotamer can be eliminated. By using rotamer pairs or more complex combinations rather than single rotamers, the efficiency of the method can be increased further [5]. Dead end elimination requires that the energy function be expressible as the sum of individual and pairwise terms, since clearly that is necessary for the above criteria to be valid. This requirement is limiting since more accurate energy functions may want to include more terms. The method also absolutely requires a discretization of the search space, in order to be able to prune, whereas it is not an essential requirement for Monte Carlo or genetic algorithms. Dead end elimination, being quasi-exhaustive and nonstochastic, will see computational performance decline as the search space becomes larger.

We can see that each search algorithm has its advantages and disadvantages. Stochastic methods scale well with increased complexity but do not guarantee convergence to a global minimum. Deterministic methods offer the guarantee of at least getting self-consistent solutions (same answer every time), and methods like dead end elimination will even offer a guarantee of finding the global minimum. Roughly speaking, the stochastic methods are faster, while deterministic methods are more accurate. For smaller design problems, the deterministic methods may be used, while for larger ones, stochastic [14,5]. An obvious question is whether a combination of the types of algorithms—perhaps one where search is initially stochastic and then, in areas found to be of interest, deterministic—might offer benefits. Pokala and Handel agree and say that this might be so [14].

SUMMARY OF DRAWBACKS AND CHALLENGES

As noted earlier, protein design methods today have key drawbacks. Among these are that the approximations used result in loss of good, low energy sequences; the methods are too slow to be used on many problems of interest; and the accuracy of energy functions in use is questionable. I group obstacles to solving these drawbacks into two main categories: problems of computational technique and power and problems of biophysical knowledge. The approximations issue falls under the former, as with an infinite amount of computing power, we would not need to make any approximations. The slowness problem is obviously under the first heading as well. The energy functions' accuracy issue is shared by both categories, as they exclude factors in their efforts to remain computationally practicable (first category), but it is an open question whether even if they did include everything they could they would faithfully represent reality (second category), since we have no way of testing that possibility.

The fixed backbone approximation was central to protein design for a long time, and goes very far in reducing computational complexity. However, it is an approximation that has a huge impact on the results achieved, and has been given as the reason why most computationally designed sequences resemble the native sequence of the starting structure [9]. Indeed, fixed backbone designed sequences average 90% identity with the sequence of the starting structure, whereas Larson reports just 50% to 60% identity in the sequences he designed using the Genome@home SPA process described above, where backbones were perturbed only up to 1 Å. Proteins naturally utilize small adjustments in their backbones to accommodate certain mutations, and the design using a fixed backbone does not allow for this [1]. Except when trying to design a new sequence that will produce a structure as close to the starting structure as possible, the approximation is really far from ideal. And for design of proteins where the exact final structure is not known (it is not very similar to known proteins), backbone flexibility is absolutely required. A few recent studies have relaxed the approximation, but in order to keep the computation manageable, they mostly either did a very coarse variation of backbone structure such as relative arrangement of secondary structure elements or designed only a portion of the target protein [9]. Increased computational power and better techniques must be developed to free us from the fixed backbone approximation and let the field progress to larger proteins and to more complex problems like design of proteins with particular functions.

This ties with the second problem of slowness. If computers were infinitely fast and powerful, then we could make models as complicated and comprehensive as we wanted. As it is though, depending on the method used, even a small protein of a few hundred amino acids can take weeks to design, and large systems are impractical for most.

Energy functions must be simple enough to be computationally tractable, but accurate enough for comparisons of sequence energies in the search to be correct. AMBER, OPLS, DREIDING, and CHARMm are among the force fields often used, taking into account terms like van der Waals, electrostatics, dihedral angle, bond angle, and bond stretching interactions. Parameters have been adjusted in an ad hoc manner to try to improve results. Greater realism in the energy functions is required for some problems. For computational design of enzymes, for instance, energy functions must more accurately deal with charged and polar groups [14], which are very important to enzyme function.

PROPOSED ENHANCEMENTS

We have seen how far protein design has come and what is preventing it from going farther to tackle the many remaining challenges. How can we carry protein design the next leg of its journey, doing justice to its short and accomplished history of innovation, and developing it to handle problems like more quickly, consistently, and generically designing small proteins; designing large (thousands of residue) proteins; designing proteins that bind particular peptides (involves "negative design" since in addition to binding the peptide we want the protein to not bind with anything else); designing substrate specific enzymes; and designing improved pharmaceuticals (which overlaps with previous problems)? While I do not by any means claim to have solutions to all the obstacles discussed, or even fully tested solutions to a few, I do have a few suggestions, some of which I myself will try out (I have not had nearly the time to complete any tests or I would include the results here).

Parallelization

The Genome@home distributed computing project has around 3000 processors (actually, far more, as noted later) available for use worldwide, giving its researchers more computational power than other protein designers have. It has used this to do things like design with flexible backbones. But even it would need more computational power, using its current methods, to use the more detailed energy function that Larson would like to use for designing protein-protein interfaces. In personal discussion with Larson, an obvious improvement seemed to be that of increasing the amount of parallelization in the calculations. The impact of this would be most dramatic on the scale of Genome@home but would be just as valid for the smaller scale computing clusters used by most designers. Parallelization would neither decrease the amount of computation a problem requires nor decrease the total amount of computational time required, but it would make usage of computing resources more efficient and thus decrease the wall time. In other words, if a given design problem requires 24 hours of processor time, it would still require that much processor time, but it might be able to actually get done in just a fraction of that of time. With processors becoming cheaper and cheaper and more and more available, the relative importance of human wall time over processor time will become more pronounced.

Using the concrete description given earlier of SPA as a basis, let me give a specific possible parallelization scheme. Currently, Genome@home gives a client machine a work unit that has it do a full design on a particular backbone. That is, a single client will do the 30 iterations of the 200 rounds of genetic recombination, each round of which includes energy calculations on 300 models. The only parallelization in this procedure, if that word is to be used, is that each client may be given a different backbone or a different seed. The energy functions Larson would like to use are "orders of magnitude" slower than what SPA currently employs,

requiring greater parallelization to maintain current times to results. There are two straightforward places to divide the calculation. One is to have a single client machine do only one of the 30 outer loop iterations. The server can then choose between the best results of the 30 calculations that 30 different machines did. Assuming the availability of an unlimited number of machines, this would give at best about a 30 times speedup over current times. For a finer division of labor, each energy calculation could be assigned to a different client machine. Recombination would be done by the server, a job the server could easily handle as the recombination itself takes only about a second (for between 300 models). Ignoring the costs of communication between client machines and the server and continuing to assume an unlimited number of client machines, this would give at most a 9000 times speedup since each round of recombination would be sped up 300 times (by having the 300 energy calculations for it occur in parallel), and this would be on top of the 30 times speedup from dividing the outer loop as described earlier.

After discussing parallelization with Larson for this paper, he has offered to let me to implement a parallelization scheme for Genome@home. We hope that it will dramatically increase the power of what can be accomplished. For SPA, about 9000 dedicated processors would be required to reap the full benefit. Genome@home currently has access to the 70,000 machines running Folding@home, another distributed computing project, so that number is not impossible to reach. One must keep in mind, though, that the no overhead assumption I made above is a major one. My guess would be that actual practice would see improvements an order of magnitude or so less than the theoretical maximum.In practice, distributed computing works best when the ratio of computation time to networking time is high, so what might work well is to, rather than try to get one protein design done as fast as possible, try to do two or three at once, so that each client can at a time get several energy calculations at once rather than just one—a medium between the current practice and the extreme of parallelization.

Even a ten times speedup will go a way towards to removing the lack of computational power obstacle standing in the way of solving many of the drawbacks and challenges discussed earlier.

Hierarchical energy functions

Whereas the previous improvement I put forward requires an enormous number of processors and so is not very generally useful, this one is. Energy functions must be computationally tractable and also accurate, two goals that have competing interests. The use of a hierarchy of energy functions may help. Initially, fast but not too accurate energy functions can be used to search space. Once areas of interest have been identified (local minima in stochastic methods like Monte Carlo or genetic algorithms, or areas left after preliminary pruning in dead end elimination), then more accurate and computation intensive functions can be used locally. I did not find literature references to anything like this, but it appears a common sense thing to try. It can be a way to allow energy functions to include the greater accuracy needed for some problems of interest without sacrificing tractability.

Use of highly precise free energies

Building on the previous idea, the more accurate energy function used on areas of search space interest need not be atomic level molecular mechanics force fields. Work is being done on highly accurate ΔG values for interactions. These techniques could be used particularly for

design of proteins intended to bind specifically with peptides or substrate. They are very computationally intensive, but used judiciously in the search space, may be made practical.

I do not wish to imply that these proposed enhancements will solve protein design or even guarantee that they will have major effect, but they do offer avenues worth exploring. Improvements to techniques are what will allow protein design to surmount the obstacles it faces to overcoming its drawbacks.

CONCLUSION

Computational protein design has come a long way in its short history. The basic standard approach, regardless of which search technique is used, must see improvements and overcome the obstacles regarding computational power and technique (for search) and energy function in order to tackle the important unsolved problems remaining. Fortunately, there are avenues of improvement wide open for examination. With some work, we can see the promise of computational protein design—understanding the relationship between sequence and structure, aiding structural and functional genomics, and creating proteins with desired functions, most significantly—fulfilled. A boom is likely to be seen, I predict, in the construction of nature's building blocks.

ACKNOWLEDGMENTS

I would like to thank Professor Douglas Brutlag for his course and for directing me to write this paper, which may lead to good future work for me. I also thank Stefan Larson for talking with me about protein design and answering my questions about SPA and Genome@home.

REFERENCES

* Those marked with an asterisk were especially useful.

[1] E.P. Baldwin, O. Hajiseyedjavadi, W.A. Baase, & B.W. Matthews. The role of backbone flexibility in the accommodation of variants that repack the core of T4 lysozyme. Science. 262(5140):1715-1718 (1993).

[2] D.N. Bolon, C.A. Voigt, & S.L. Mayo. De novo design of biocatalysts. Current Opinion in Chem.Bio. 6:125-129 (2002).

[3] B.I. Dahiyat & S.L. Mayo. De Novo Protein Design: Fully Automated Sequence Selection. Science. 278:82-87 (1997).

[4] William F. DeGrado. Proteins from Scratch. Science. 278(5335):80-81 (1997).

[5] *J.R. Desjarlais & N.D. Clarke. Computer search algorithms in protein modification and design. Curr Opin Struct Biol. 8(4):471-475 (1998).

[6] K.E. Drexler. Molecular engineering: an approach to the development of general capabilities for molecular manipulation. Proc. Natl. Acad. Sci. USA, 78:5275-5278 (1981).

[7] Ho and W. F. DeGrado. Design of a 4-Helix bundle protein: Synthesis of peptides which self-associate into a helical protein. J. Am. Chem. Soc. 109: 6751-6758 (1987)

[8] W.L. Jorgensen & J. Tirado-Rives. The OPLS potential functions for proteins. Energy minimizations for crystals of cyclic peptides and crambin. J Am Chem Soc. 110:1657-1666 (1988).

[9] *S.M. Larson, J.L. England, J.R. Desjarlais, V.S. Pande. "Thoroughly sampling sequence space: large-scale protein design of structural ensembles." Protein Science. 11(12):2804-2813 (2002).

[10] *Stefan Larson. Personal conversation on November 19, 2002.

[11] B. W. Matthews, H. Nicholson, & W. J. Becktel. Enhanced protein thermostability from site-directed mutations that decrease the entropy of unfolding. Proc. Nat. Acad. Sci. 84:6663-6667 (1987).

[12] C. Pabo. Molecular technology. Designing proteins and peptides. Nature. 301(5897):200 (1983).

[13] L. J. Perry and R. Wetzel. Disulfide bond engineered into T4 lysozyme: stabilization of the protein toward thermal inactivation. Science. 226:555-557 (1984).

[14] *Navin Pokala and Tracy Handel. Review: Protein Design—Where We Were, Where We Are, Where We're Going. J. of Struct. Bio. 134: 269-281 (2001).

[15] K. Raha, A.M. Wollacott, M.J. Italia, & J.R. Desjarlais. Prediction of amino acid sequence from structure. Protein Sci. 9(6):1106-19 (2000).

[16] L. Regan and W. F. DeGrado. Characterization of a helical protein designed from first principles. Science. 241:976-978 (1988).

[17] C.A. Voigt, S.L. Mayo, F.H. Arnold, & Z.G. Wang. Computational method to reduce the search space for directed protein evolution. Proc Natl Acad Sci U S A. 98(7):3778-83 (2001).

[18] S.J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, C. Ghio, G. Alagona, S. Profeta, & P. Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. J AmChem Soc. 106:765-784 (1984).