Protein Engineering vol.15 no.10 pp.779–782, 2002

Protein Design is *NP*-hard

Niles A.Pierce[1,2] and Erik Winfree[3]

[1]Applied and Computational Mathematics and [3]Computer Science and Computation and Neural Systems,California Institute of Technology, Pasadena, CA 91125, USA

[2]To whom correspondence should be addressed.
E-mail: niles@caltech.edu

Biologists working in the area of computational protein design have never doubted the seriousness of the algorithmic challenges that face them in attempting *in silico* sequence selection. It turns out that in the language of the computer science community, this discrete optimization problem is *NP*-hard. The purpose of this paper is to explain the context of this observation, to provide a simple illustrative proof and to discuss the implications for future progress on algorithms for computational protein design.
Keywords: complexity/design/*NP*-complete/*NP*-hard/proteins

Introduction

The protein design problem may be formulated in many different ways; here, we focus on a simple definition that has gained significant attention (Desjarlais and Handel, 1995; Dahiyat and Mayo, 1996, 1997; Gordon and Mayo, 1998; Malakauskas and Mayo, 1998; Koehl and Levitt, 1999; Pierce *et al.*, 2000; Shimaoka *et al.*, 2000; Wernisch *et al.*, 2000; Looger and Hellinga, 2001; Gordon *et al.*, 2002). The objective is to optimize the stability of a specified backbone fold that is assumed to be rigid. At each residue position in the design, different amino acid alternatives are compared in the form of discrete side-chain 'rotamers', representing the statistically dominant orientations of amino acid side chains in naturally occurring proteins (Janin *et al*., 1978; Ponder and Richards, 1987; Dunbrack and Karplus, 1993). Choosing one rotamer at each position defines the global conformation of all the atoms in the system and implicitly specifies an amino acid sequence. Different conformations are ranked using an empirical potential function that attempts to quantify the free energy of the system. For simplicity, the potential function is assumed to contain only pairwise terms, which may be used to describe van der Waals, electrostatic and hydrogen bonding interactions, as well as solvent exposure (Gordon *et al.*, 1999). [Rotamer self-energies that are typically used in protein design to capture the interactions with the backbone can be folded into the pairwise terms (Gordon and Mayo, 1999).] In formulating the protein design problem, we intentionally use intuitive (but technically imprecise) notation in order to reach the widest audience. Computer scientists should have little difficulty filling in the details.

Protein design (PRODES) can be described in *optimization form* as (see Figure 1):

Given p disjoint sets of rotamers R_i (one set for each position i) and a potential function $E(\cdot,\cdot)$ that returns the energy between a pair of rotamers at different positions, choose the rotamer $r_i \in R_i$ at each position that minimizes the sum of the pairwise interaction energies between all positions:

$$E_{\text{total}} \equiv \sum_i \sum_{j,\, j<i} E(r_i, r_j)$$

A solution to this problem is called a global minimum energy conformation (GMEC). There is currently no known algorithm for identifying a GMEC solution efficiently (in a specific sense to be defined shortly). Given the failure to identify such an approach, it is worth attempting to discern whether this is even a reasonable goal. Fortunately, a beautiful theory from computer science allows us to classify the tractability of our problem in terms of other discrete optimization problems (Garey and Johnson, 1979; Papadimitriou and Steiglitz, 1982). This theory does not apply directly to the optimization form, but instead to a related *decision form* that has a 'yes/no' answer. The decision form of PRODES becomes:

Is there a rotamer at each position such that $E_{total} \leqslant K$ for a specified constant K?

In complexity theory, a *problem* is formally a set of input–output pairs. A particular input is called an *instance* of the problem. An algorithm *solves* a problem if it produces the correct output for every valid instance. To classify algorithm efficiency, it is useful to define the concepts of *polynomial-time* and *polynomial-space* algorithms. These imply, respectively, that the algorithm requires a number of steps and an amount of data storage space that are at most polynomials of the length of the data describing the instance (e.g. if $|x|$ is the length of the input data x, then there exist constants α and β such that there are polynomial bounds $\alpha|x|^{\beta}$, as opposed to exponential bounds $\alpha\beta^{|x|}$).

The PRODES decision problem can now be identified with one or more of the following three complexity classes:

P: Contains problems for which polynomial-time algorithms are known to exist.

NP: Contains problems for which polynomial-time algorithms are not necessarily known. The requirement is that for every 'yes' instance x of a problem A, there must exist a polynomial-space *certificate* for x that can be checked for validity in polynomial-time.

NP-complete: Contains those problems in *NP* to which all other problems in *NP* can be polynomial-time reduced. (A problem A in *NP* is polynomial-time reducible to a problem A' in *NP*, if given a string x, a string x' can be constructed in polynomial-time such that x is a 'yes' instance of A if and only if x' is a 'yes' instance of A'.)

Problems in class *P* are considered *easy* and the corresponding algorithms are considered *efficient*. If problem A' is easy and problem A is polynomial-time reducible to A', we can conclude that A is also easy. A longstanding conjecture in complexity theory is that $P \neq NP$ and in particular, that no

polynomial-time algorithms exist for problems that are *NP*-complete. However, *NP*-complete problems do have the significant property that if there is a polynomial-time algorithm for *any NP*-complete problem, then there is a polynomial-time algorithm for *all* problems in *NP*. Ideally we would like to discover a polynomial-time algorithm for PRODES. However, before expending further effort in this endeavor, it is worthwhile first to attempt to determine whether this problem is *NP*-complete. If so, decades of failed attempts to find a polynomial-time algorithm for *any NP*-complete problem suggest that there is little cause for optimism.

Materials and methods

To prove that the decision form of PRODES is *NP*-complete, we must show:

Step 1: that PRODES belongs to the class *NP*.

Step 2: that all other problems in *NP* polynomial-time reduce to PRODES.

From the definition of *NP*-completeness, this second requirement can be satisfied by polynomial-time reducing any *one* known *NP*-complete problem to PRODES.

One famous *NP*-complete problem is *satisfiability* (SAT), an important problem in mathematical logic:

Given M disjunctive clauses C_1, C_2, ..., C_M involving N Boolean (T/F) variables x_1, x_2, ..., x_N, is the formula $C_1 \cdot C_2 \ldots \cdot C_M$ satisfiable?

For example, the instance $(x_1 + \bar{x}_2 + x_3)\cdot(\bar{x}_1 + x_2 + x_3)\cdot\bar{x}_3$, with $M = 3$ and $N = 3$, is satisfiable and has the two valid certificates $(x_1,x_2,x_3) \in \{(T,T,F),(F,F,F)\}$. Here, '·' represents 'and', '+' represents 'or' and 'x̄' represents the negation of x. Variables and their negations (e.g. x_k and $\bar{x}_k$) are called *literals*. In general, SAT can be solved in exponential-time by trying all possible truth assignments, requiring up to 2^N tests. It is remarkable that there is no known polynomial-time algorithm for this simple problem. Quite to the contrary, it is the first problem to have been proved to be *NP*-complete (Cook, 1971; Karp, 1972; Levin, 1973) and often serves as the starting point for other *NP*-completeness proofs.

Results

We now seek to prove that the decision form of PRODES is *NP*-complete by considering two steps in turn:

Step 1: PRODES belongs to *NP* since for a 'yes' instance, the validity of a polynomial-space certificate consisting of p rotamer indices can be verified in polynomial-time by evaluating E_{total}.

Step 2: We will reduce SAT to PRODES. Consider a SAT instance with M clauses and N variables, where clause i has L_i literals. Construct PRODES with $p = M + N$ positions and $K = 0$ (e.g. see Figure 2):

M positions, each representing a clause i, each with L_i rotamers representing the literals in clause i;

N positions, each representing a variable x_k, each with two rotamers representing the possible values T or F for x_k.

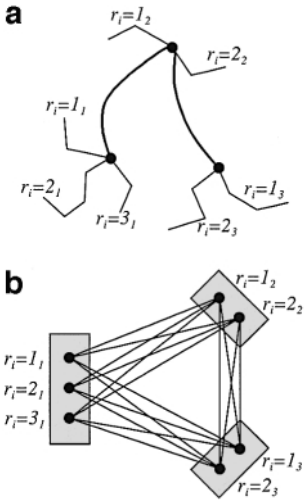

Fig. 1. (**a**) Representative protein backbone with three rotamers at position $i = 1$ and two rotamers at positions $i = 2$ and $i = 3$. (**b**) The corresponding graph. Each box represents a backbone position; each circular node represents a rotamer alternative; each edge connecting two nodes is weighted by the pairwise energy. In this picture, the goal is to choose the one node within each box that minimizes the sum of the three edge weights connecting the nodes.

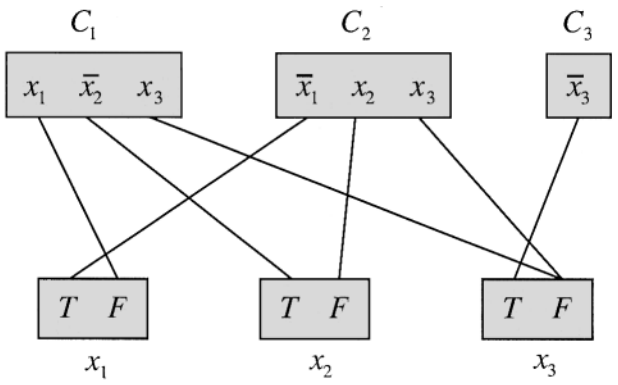

Fig. 2. Reduction of SAT into PRODES for the instance $(x_1 + \bar{x}_2 + x_3)\cdot(\bar{x}_1 + x_2 + x_3)\cdot\bar{x}_3$. Only edges between rotamers with energy $E(\cdot,\cdot) = 1$ are depicted. The problem is satisfiable if and only if rotamers can be selected that exclude all of these non-zero pairs. For this 'yes' instance, the two valid certificates are $(x_1,x_2,x_3) \in \{(T,T,F),(F,F,F)\}$.

Now assign the following energies to each of the rotamer pairs:

$E(r_i,r_j) = 1$ if r_i represents the literal x_k (a non-negated variable) and r_j represents the value F for x_k;

$E(r_i,r_j) = 1$ if r_i represents the literal $\bar{x}_k$ (a negated variable) and r_j represents the value T for x_k;

$E(r_i,r_j) = 0$ otherwise.

With this specification, it is possible to select rotamers for the PRODES instance such that $E_{\text{total}} = 0 \leqslant K$ if and only if the original SAT instance is satisfiable. That is, a valid certificate for a 'yes' instance of PRODES can be used to construct a validate certificate for a 'yes' instance of SAT and vice versa. In total, there are fewer than $3MN$ rotamers, so that $E(\cdot,\cdot)$ can be contained within a $3MN \times 3MN$ matrix. Hence, a polynomial-space instance of PRODES can be constructed in polynomial-time given an instance of SAT, and the instance will be 'yes' for PRODES if and only if it is 'yes' for SAT.

This proves that the decision form of PRODES is *NP*-

complete. [We chose to show that PRODES is *NP*-complete by reducing SAT to PRODES. Other known *NP*-complete problems also naturally reduce to PRODES. Graph coloring (Garey and Johnson, 1979) is *NP*-complete (Karp, 1972) and can be seen as a strict subset of PRODES (J.Hartline, personal communication, 2000), wherein each position (node in the graph) has the same number of rotamers (colors) and the energy function is unity for edges joining rotamers of the same color and zero otherwise. Maximum *a posteriori* inference in graphical belief networks (Pearl, 1988) is *NP*-complete (Shimony, 1994) and the specific case of pairwise Markov nets (Weiss, 2000) is formally identical with PRODES. In this case, the positions correspond to variables, the rotamers represent the possible states of each variable and the energy function quantifies the conditional probabilities between these states. Consequently, the considerable efforts devoted to developing algorithms for one problem may in some cases be directly applied to other problems.]

Discussion

So is it good news or bad news to discover that the decision form of PRODES is *NP*-complete? Certainly, it would be more useful for ongoing design efforts to be able to show that PRODES belongs to the class *P,* for then we would have a polynomial-time algorithm for solving the problem (at least in decision form). However, proving that PRODES is *NP*-complete provides critical information: we now know that protein design is as hard as any problem in the class *NP*, including such celebrated problems as the traveling salesman problem. This implies that past failures in developing efficient algorithms for all other *NP*-complete problems now apply also to PRODES. Alternatively, PRODES now joins the list of problems for which the discovery of a polynomial-time algorithm would revolutionize the field of complexity theory.

An optimization problem for which the related decision problem is *NP*-complete is termed *NP*-hard. Hence, the optimization form of PRODES is *NP*-hard. More generally, the class '*NP*-hard' is comprised of those problems, in any form, for which a polynomial-time algorithm could be used to provide a polynomial-time algorithm for an *NP*-complete problem. Such problems are at least as hard as any problem in *NP.*

It is important to keep in mind that the classification of protein design as an *NP*-hard optimization problem is a reflection of worst-case behavior. In practice, it is possible for an exponential-time algorithm to perform well or for an approximate stochastic method to prove capable of finding excellent solutions to *NP*-complete and *NP*-hard problems. This is the case for stochastic local search algorithms for SAT (Selman *et al.*, 1992, 1994), which are creating excitement in the computer science community because they can dramatically outperform all known deterministic algorithms for this problem (Hoos and Stutzle, 2000). In the protein design community, stochastic methods based on Monte Carlo, simulated annealing or genetic algorithms have performed with some success on small protein design problems (Desjarlais and Clarke, 1998), but deviate substantially from the GMEC solution as the problem size increases (Voigt *et al.*, 2000). Deterministic exponential-time methods such as dead-end elimination (Desmet *et al.*, 1992; Goldstein 1994; Gordon and Mayo 1998; Pierce *et al.*, 2000; Looger and Hellinga, 2001; Gordon *et al.*, 2002) and branch and bound (Leach and Lemon, 1998; Gordon and Mayo, 1999; Wernisch *et al.*, 2000) are guaranteed to converge to the GMEC solution when they do converge. The key question is how these algorithms perform on problems of interest to the protein design community.

Algorithm performance varies widely with the physical model (Gordon *et al.*, 2002), as characterized by the empirical potential function and the rotamer library size. However, at present, with an experimentally validated physical model and an exact search method, it is often possible to design entire core, boundary or surface regions of small protein domains (representing ~25–75 residues at a time) (Gordon *et al.*, 2002). Ultimately, it would be desirable to obtain GMEC solutions for entire domains with 100–200 residues. Note that as the number of residues increases from 75 to 200, the number of conformations increases from $O(n^{75})$ to $O(n^{200})$, where n is the average number of rotamers per position.

How should the research community attempt to bridge this sizeable gap? The knowledge that the problem is *NP*-hard implies that we may always have to rely on exponential-time algorithms if GMEC solutions are required. Hence, current state-of-the-art methods that have exponential worst-case bounds continue to merit further research. These methods are able to identify global minima for problems that would require 10^{70} times the age of the universe using an exhaustive search. Similar further improvements would dramatically increase the scope of rational protein design. If such advancements prove elusive using exact algorithms, then improved stochastic, heuristic or approximate methods will likely play an increasingly important role in protein design. The finding that protein design optimization is *NP*-hard thus reinforces the need for continued efforts on exact exponential-time and approximate stochastic methods, encourages interaction with other scientific communities working on *NP*-hard optimization problems and drastically increases the pessimism for finding an efficient polynomial-time algorithm.

It remains possible that the protein design problem can be simplified (e.g. by placing restrictions on the potential function) to yield a less general problem for which polynomial-time algorithms can be devised. [It is instructive to consider the situation with variants of the traveling salesman problem (TSP)—perhaps the most-studied *NP*-hard problem (Lawler *et al.*, 1985). TSP cannot even be efficiently approximated to within any constant factor (Sahni and Gonzalez, 1976). However, when restricted to the important subclass of Euclidean constant-dimension problems, finding exact optimal solutions remains *NP*-complete, but approximate TSP is dramatically easier (nearly linear-time for randomized algorithms) (Arora, 1998). Thus important open questions are to determine whether PRODES is efficiently approximable and to find natural restrictions on PRODES that allow for more efficient algorithms.] It may also be possible that there exist alternative formulations of the protein design problem that are of equal intellectual merit and practical utility (e.g. design for surface shape or chemistry with no restriction on the backbone fold), but greater computational tractability. These alternative formulations are also likely to be *NP*-hard, so it remains an important open question as to whether the protein design problem can be described in a form that is both computationally tractable and biologically sound.

Acknowledgements

We thank L.J.Schulman for a critical reading of the manuscript. This research was supported by the Burroughs-Wellcome Foundation through the Caltech Initiative in Computational Molecular Biology (NAP) and by the Defense

Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under agreement F30602-010200561 (both authors).

References

Arora,S. (1998) *J. Assoc. Comput. Machin.*, **45**, 753–782.
Cook,S.A. 1971. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*. Association of Computing Machinery, New York, pp. 151–158..
Dahiyat,B.I. and Mayo,S.L. (1996) *Protein Sci.*, **5**, 895–903.
Dahiyat,B.I. and Mayo,S.L. (1997) *Science*, **278**, 82–87.
Desjarlais,J.R. and Clarke,N.D. (1998) *Curr. Opin. Struct. Biol.*, **8**, 471–475.
Desjarlais,J.R. and Handel,T.M. (1995) *Protein Sci.*, **4**, 2006–2018.
Desmet,J., De Maeyer,M., Hazes,B. and Lasters,I. (1992) *Nature*, **356**, 539–542.
Dunbrack,R.L.,Jr and Karplus,M. (1993) *J. Mol. Biol.*, **230**, 543–574.
Garey,M.R. and Johnson,D.S. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, New York.
Goldstein,R.F. (1994) *Biophys. J.*, **66**, 1335–1340.
Gordon,D.B. and Mayo,S.L. (1998) *J. Comput. Chem.*, **19**, 1505–1514.
Gordon,D.B. and Mayo,S.L. (1999) *Structure*, **7**, 1089–1098.
Gordon,D.B., Marshall,S.A. and Mayo,S.L. (1999) *Curr. Opin. Stuct. Biol.*, **9**, 509–513.
Gordon,D.B., Hom,G.K., Mayo,S.L. and Pierce,N.A. (2003) *J. Comput. Chem.*, **24 (2)**, to appear.
Hoos,H.H. and Stutzle,T. (2000) *J. Autom. Reasoning*, **24**, 421–481.
Janin,J., Wodak,S., Levitt,M. and Maigret,D. (1978) *J. Mol. Biol.*, **125**, 357–386.
Karp,R.M. (1972) In Miller,R.E. and Thatcher,J.W. (eds), *Complexity of Computer Computations*. Plenum Press, New York, pp. 85–103.
Koehl,P. and Levitt,M. (1999) *J. Mol. Biol.*, **293**, 1161–1181.
Lawler,E.L., Lenstra,J.K., Rinnooy Kan,A.H.G. and Shmoys,D.B. (1985) *The Traveling Salesman Problem*. Wiley, New York.
Leach,A.R. and Lemon,A.P. (1998) *Proteins*, **33**, 227–239.
Levin,L.A. (1973) *Probl. Inf. Transmiss.*, **9**, 265–266.
Looger,L.L. and Hellinga,H.W. (2001) *J. Mol. Biol.*, **307**, 429–445.
Malakauskas,S.M. and Mayo,S.L. (1998) *Nature Struct. Biol.*, **5**, 470–475.
Papadimitriou,C.H. and Steiglitz,K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ.
Pearl,J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kauffman, San Francisco.
Pierce,N.A., Spriet,J.A., Desmet,J. and Mayo,S.L. (2000) *J. Comput. Chem.*, **21**, 999–1009.
Ponder,J.W. and Richards,F.M. (1987) *J. Mol. Biol.*, **193**, 775–791.
Sahni,S. and Gonzalez,T. (1976) *J. Assoc. Comput. Machin.*, **23**, 555–565.
Selman,B., Levesque,H. and Mitchell,D. (1992) In *Proceedings of AAAI '92*. MIT Press, Cambridge, MA, pp. 440–446.
Selman,B., Kautz,H.A. and Cohen,B. (1994) In *Proceedings of AAAI '94*. MIT Press, Cambridge, MA, pp. 337–343.
Shimaoka,M., Shifman,J.M., Jing,H., Takagi,J., Mayo,S.L. and Springer,T.A. (2000) *Nature Struct. Biol.*, **7**, 674–678.
Shimony,S.E. (1994) *Artif. Intell.*, **68**, 399–410.
Voigt,C.A., Gordon,D.B. and Mayo,S.L. (2000) *J. Mol. Biol.*, **299**, 789–803.
Weiss,Y. (2000) *Neural Comput.*, **12**, 1–41.
Wernisch,L., Hery,S. and Wodak,S.J. (2000) *J. Mol. Biol.*, **301**, 713–736.

Received March 13, 2002; revised May 31, 2002; accepted July 2, 2002