John von Neumann Institute for Computing

Sequential Monte Carlo Methods for Protein Folding

Peter Grassberger

published in

NIC Symposium 2004, Proceedings,
Dietrich Wolf, Gernot Münster, Manfred Kremer (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **20**, ISBN 3-00-012372-5, pp. 1-10, 2003.

http://www.fz-juelich.de/nic-series/volume20

Sequential Monte Carlo Methods for Protein Folding

Peter Grassberger

Complex Systems Research Group
John von Neumann Institute for Computing
Research Centre Jülich, 52425 Jülich, Germany
E-mail: p.grassberger@fz-juelich.de

We describe a class of growth algorithms for finding low energy states of heteropolymers. These polymers form toy models for proteins, and the hope is that similar methods will ultimately be useful for finding native states of real proteins from heuristic or a priori determined force fields. These algorithms share with standard Markov chain Monte Carlo methods that they generate Gibbs-Boltzmann distributions, but they are not based on the strategy that this distribution is obtained as stationary state of a suitably constructed Markov chain. Rather, they are based on growing the polymer by successively adding individual particles, guiding the growth towards configurations with lower energies, and using "population control" to eliminate bad configurations and increase the number of "good ones". This is not done via a breadth-first implementation as in genetic algorithms, but depth-first via recursive backtracking. As seen from various benchmark tests, the resulting algorithms are extremely efficient for lattice models, and are still competitive with other methods for simple off-lattice models.

1 Introduction

Research in modern biology is more and more based on heavy numerical computations. This includes the reconstruction of complex structures from X-ray imaging, sequence alignment, the optimization of phylogenetic trees, simulation of protein networks, and – last but not least – protein folding. The latter is considered by many as one of the most challenging problems in mathematical biology. It is concerned with the problem of how a heteropolymer of a given sequence of amino acids folds into precisely that geometric shape in which it performs its biological function[1–4]. Currently, it is much simpler to find coding DNA — and, thus, also amino acid — sequences than to elucidate the 3-d structures of given proteins. Therefore, solving the protein folding problem would be a major break-through in understanding the biochemistry of the cell.

Indeed, there are several closely related problems associated with protein folding. What we have alluded to above is the *fold prediction* problem. In addition there is the inverse fold prediction problem, concerned with designing a sequence which has a given 3-d structure as its native state. This is obviously a central problem in ab initio drug design, and in designing artificial proteins. Again another problem is that of discovering the detailed pathways followed during the folding process. This is obviously important if one wants to interfere during it, as would be useful in treatments of cancer, BSE, and Alzheimer.

In the present review we shall only be concerned with the direct native state prediction problem. At present, the most successful methods are not based on physico-chemical principles like force fields (potential energies), but on exploiting analogies with similar proteins whose structure has been determined by X-ray or NMR studies. This data-based approach will stay the method of choice for some time, but it cannot fully substitute a more fundamental approach where one starts from a force field and identifies the native state as the state with lowest free energy. The basis of such an approach are of the course

the famous experiments of Anfinsen who showed some thirty years ago that native states are uniquely defined by their amino acid sequences, and that not too large proteins fold spontaneously and reversibly.

In such a direct ab initio approach there are two main difficulties. The first is that presently available force fields are most likely too crude for any but the most simple proteins. These fields are typically obtained by heuristic arguments and involve drastic simplifications (such as using two-body potentials of Lennard-Jones type, fixed bond lengths, etc.). Proteins typically are not very tightly collapsed (after all, most of them are catalyzers which have to be flexible in order to function), and the differences in energy between the native and misfolded states are typically as large or even smaller than the uncertainty of the potential energies.

The other problem is that algorithms for finding low energy states of a complex potential are still too slow. There has been a dramatic improvement during recent years in the efficiency of Monte Carlo algorithms[5–9], and presently available algorithms are fast enough to find the native states of small proteins (with up to 30 amino acids or more) in vacuum. But they are still not efficient for proteins in water (and simplified treatments of the effect of water are problematic[10]), so that the method of choice today still is molecular dynamics, even though it is by far too slow to study larger proteins. Notice that fast minimization algorithms would also be extremely useful in the search for better force fields.

2 Chain Growth Methods

In view of this situation, we shall discuss in the present review the potential use of an alternative class of Monte Carlo methods which are not based on the Metropolis concept of putting up a Markov chain (with proposed moves and subsequent acceptance or rejection) whose stationary state is the wanted Gibbs-Boltzmann distribution. Instead, they are based on *sequential sampling* or polymer *growth*. Starting with an empty configuration, particle after particle is added in a biased way to take into account steric and energetic effects. This bias is compensated by a weight factor which multiplies the Boltzmann factor for the potential energy of the new particle in the field of the previous ones. The product $W_n = \prod_{i=1}^{n} w_i$ of these factors (one for each newly added particle i) gives the total weight of a configuration containing n particles. The power of such methods for generating valid configurations (not necessarily optimal ones) is discussed in Ref. 11.

If the bias is not perfect, i.e. if the bias correcting factor does not compensate perfectly the Boltzmann factor, the total weight W_n will fluctuate more and more as particles are added. As in genetic algorithms[12] and as in diffusion type quantum Monte Carlo algorithms[13], configurations with low weight are killed while "good", i.e. high weight configurations are split into two, with the weight shared among them. In order to preserve the correct statistics during this "population control", the killing is not done unconditionally but with probability 1/2: Low-weight configurations are randomly either killed or kept, in which case their weight is doubled.

Algorithms of this type were first developed for neutron transport theory in the 1950's, when they were called "Russian roulette and splitting"[14]. For recent reviews see Ref. 15,16. They were re-invented several times, and applied to proteins first by Velikson *et al.*[17]. For a recent paper see Ref. 18. In all these works, the population control was made "breadth first" as in genetic algorithms and in diffusion quantum Monte Carlo: A large number

(ca. 100 or more) of replicas are treated simultaneously, and the "goodness" of a configuration is determined by comparing it with its brothers. The advantage of this is conceptual simplicity, and the fact that the population will never explode. Also, it is ideally suited for massively parallel machines where each node handles one or a small number of replicas. The drawbacks are that typical implementations might not be completely free of bias, that it needs substantial memory space, and, if population control is frequent, it may lead to heavy communication.

An alternative is depth first implementation[19]. There one keeps only one replica at each moment, and puts markers at times when one decides for branching. The second branch is persued only when the present branch is abandoned, either because the full length of the polymer is reached or because it is killed. This backtracking is implemented most easily by recursive function calls: Each addition of a particle involves a call of a subroutine, and branching into k branches corresponds to k calls immediately following each other. While this is very compact and elegant, the main problem is that it is not immediately obvious how to decide on when to kill or branch. The solution is given by the observation that the average weight is an unbiased estimator of the partition sum (for an elementary discussion see the appendix of Ref. 20),

$$\hat{Z}_n = \frac{1}{M} \sum_{\alpha=1}^{M} W_n^{(\alpha)} \ , \tag{1}$$

where $\alpha = 1, 2, \ldots$ denotes the trials. One kills (branches) if the current weight is smaller (larger) by a given factor than the average weight of all previous configurations, i.e. than the estimated partition sum. The precise value of the factor is not important, values between 2 and 10 are usually good. After the first configurations the estimated $\hat{Z}_n$ is typically too small, whence the algorithm tends to accept too many configurations and makes too many branchings. This might lead to explosions of the population size, and tricks to avoid this are discussed in Ref. 21, 22.

As we said, we used in our work the simplest possible rule for killing: low weight configurations are always killed with probability 1/2, no matter how bad they are. There are much more sophisticated rules discussed in the literature[15], but we found them not to be necessary. It simply does not matter much if a bad configuration is kept for a few more iterations before it is finally killed. On the other hand, we found that the details of the branching process are crucial. In our first papers[21–23] we made *exact clones* in each branching event. At very low temperatures, where Boltzmann factors are large, "good" configurations can have very large weights and splitting them into two branches might not be sufficient. Thus we allowed also branching into 3 or more copies. While this was efficient in keeping the weights within narrow bounds, it was however not fully successful: Branching is of course done in the hope that different copies will follow different paths, thus leading to more diversity in the sample than without branching. While this was true at high temperatures, it was not quite true at temperatures where the native state can be reached. There, it might happen that one path has a weight much larger than all others, and thus all branches will follow this single path, leading to a loss of diversity. The same problem is encountered in genetic algorithms if one is too greedy.

The way out proposed in Ref. 20 is to make *different* first steps for each branch. Thus one first estimates (maybe crudely) the weight after the next adding of a monomer. Depending on this one decides whether one wants to kill, continue, or branch – and into how

many branches. Here one also checks that k different steps are indeed sterically possible, if one wants to make k branches. After that decision is taken, one selects one from all possible k-tuples of steps, and continues with these k branches. Details of this, including the general problem of selecting random *tuples* in Monte Carlo simulations, are given in Ref. 20.

The above strategy is straightforwardly implemented in lattice models where the number of a priori possible continuations (the number of "candidates") at each step is given by the coordination number. To implement it for off-lattice models one first has to select a random list of candidates (this is done independently at each step), before one selects tuples from this list[24]. For polymers above and near the Θ collapse temperature the optimal number of candidates is ≈ 4[24]. For ground state searches it turned out to be much larger, typically 20 to 50[25].

The algorithm with identical clones was called PERM (pruned-enriched Rosenbluth method) in Ref. 24, the version with branches forced to be different was called nPERM (new PERM) in Ref. 20. A further advantage of the latter is that it crosses smoothly over to exact enumeration, if the branching threshold is lowered to zero. Thus one can e.g. make simulations where one follows all possible paths during the first steps, and makes random samplings only for the later steps.

Results obtained with nPERM will be discussed in the next two sections. In Sec. 3 I will treat lattice models, while off-lattice models will be the subject of Sec. 4.

3 Lattice Models

3.1 The HP - Model

The most popular lattice model for protein folding is the HP model of K. Dill and coworkers[26–28]. It assumes that hydrophobicity is the main driving force in folding, and that all amino acids can be classified into just two groups: Hydrophobic ones ("H") which avoid contacts with water and thus form the core of a folded protein, and polar ones ("P") which want to be on the surface. This simplification is merged with the assumption that amino acids sit at the vertices of a regular lattice (square for 2-d, simple cubic or FCC for 3-d), to form a class of extremely minimalistic models. Proteins are modelled in this model by self avoiding walks with attractive interactions between neighbouring non-bonded H-H pairs: The energy between two hydrophobic monomers is -1 unit, while there is zero contact energy between P-P and H-P pairs.

The HP model has been justly criticized for being too much simplified[29], but it might nevertheless be very useful as a test bed for folding algorithms. In particular this is true because exact ground states can be obtained by combinatorial methods[30–32] for medium long chains, so one can verify in many cases whether the ground state is actually reached. We should point out that these exact methods work *only* for the HP model and assume the existence of a well-formed hydrophobic core (in the absence of which they don't give wrong results but simply don't converge), while our Monte Carlo method is in principle a blind general purpose method.

There are a number of benchmark test cases which were discussed thoroughly in Ref. 20. In the following we shall touch only some of them, and shall instead concentrate on examples which were overlooked in Ref. 20.

(a) A set of ten 48-mers on the SC lattice was studied in Ref. 33. This was meant as a test for the ability of conventional Monte Carlo search methods, and these methods failed

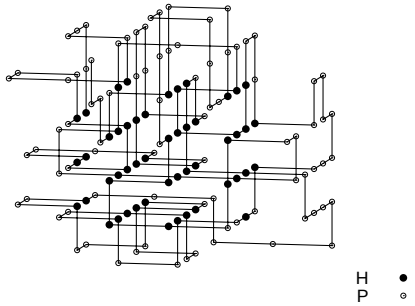

Figure 1. Configuration with $E = -80$ of a $N = 136$ HP sequence modeling a staphylococcal nuclease fragment.

in the majority of cases. With old PERM we could already fold all ten, with new PERM the CPU times were further cut down by 1 to 2 orders of magnitude to typically 10 seconds to 10 minutes on a fast PC[20].

(b) A 85-mer 2-d HP sequence was given in Ref. 34, where it was claimed to have $E_{\min} = -52$. Using a genetic algorithm, the authors could find only conformations with $E \geq -47$. In Ref. 35, using a newly developed *evolutionary Monte Carlo* (EMC) method, the authors found the putative ground state only when assuming large parts of its known structure as constraints. Within ca. 1 minute on a fast PC, we found states with $E = -53$.

(c) Four 3-d HP sequences with lengths between $N = 58$ and $N = 136$ were proposed in Ref. 36, 37 as models for actual proteins or protein fragments. Lowest energy states for these sequences were searched in Ref. 38 using a newly developed and supposedly very efficient algorithm. With nPERM, we now found states with lower energy after only few minutes CPU time, for all four chains. For the longest one, the lowest energy found by us is 15 units lower than the supposed ground state energy found in Ref. 38, but we are not sure that we did hit the true ground state. The chain is too long for the exact algorithm of Ref. 30,31. But Fig. 1 shows that our lowest energy state has a rather compact hydrophobic core, so it could well be a ground state.

(d) Ten random 64-mers on the SC lattice were chosen as tests in Ref. 39 for a genetic algorithm. This algorithm was criticized by Patton *et al.*[40] as being not optimal and too much influenced by Monte Carlo philosophy. Indeed, with a supposedly optimal genetic algorithm Patton *et al.* found in most cases lower energy states, by up to 5 units. Unfortunately no timings were given in Ref. 39, 40. With nPERM we found in *all* cases even lower energies than Patton *et al.*, by up to 8 units (see Table 1). Our energies are rather close to lower bounds obtained by estimating the hydrophobic core, so some of them might be real ground states. While reaching some of the lowest energies needed up to 8 hours on a 2 GHz PC, others were reached within seconds (in which case it was checked that no lower energies were reached even after a day of CPU time). All energies reached in Ref. 40 would have been reached within less than half a second.

(e) In these simulations we have tried to let the chains grow from either end. Sometimes growing from one end is more successful than growing from the other. This is even more pronounced in some other sequences, one example of which is shown in Fig.2. This se-

sequence Nr.	Ref. 39	Ref. 40	nPERM	lower bounds
1	-27	-27	-32	-33
2	-29	-30	-38	-41
3	-35	-38	-45	-49
4	-34	-34	-42	-45
5	-32	-36	-43	-48
6	-29	-31	-35	-37
7	-20	-25	-28	-32
8	-29	-34	-38	-43
9	-32	-33	-41	-45
10	-24	-26	-31	-35

Table 1. Lowest energies of the 10 3-dimensional HP sequences of Unger *et al.*, see paragraph (d).

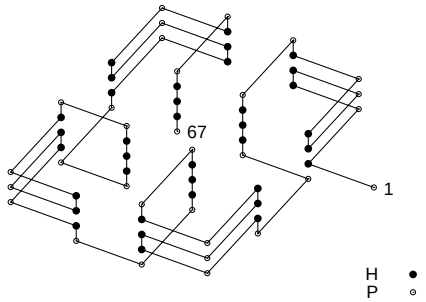

Figure 2. Ground state configuration ($E = -56$) of the $N = 67$ HP sequence given by Yue and Dill. It forms a structure resembling an α/β barrel. When starting at monomer #67 (β sheet end), nPERM could find it easily, but not when starting from monomer #1.

quence was devised by Yue and Dill[31] such that the α/β barrel shown in Fig.2 is provably the ground state. When we tried to find this configuration by starting with the monomer #1, nPERM did not succeed, because one encounters repeatedly structures which are very unstable during the growth phase. They would become stabilized later when more monomers are added, but before that could happen the configuration is killed as being not promising. What is lacking in PERM (as in any growth algorithm known to us) is an efficient way to look sufficiently far ahead so that bottlenecks formed by configurations looking bad to a short-sighted eye can be overcome. Of course one can make look-aheads where one scans all configurations up to a few (typically 2 to 5) steps ahead[18], but we found this in general to be too time consuming for being efficient.

(f) Some of the lowest energy states shown in Table 1 were obtained by a further trick. When placing a polar (P) monomer next to an H, one might block a site which would be needed during later stages of the growth. Thus it is advantageous to disfavour the contact between H and P, even though such contacts are energetically neutral. Formally this is done

by adding a repulsive P-H potential (of ≈ 0.2 units) which is included in the Boltzmann factors but which is of course not included in the final energy count. This means that folding becomes easier if one keeps the chain *less compact* during the growth. It seems to disagree with claims to the contrary, albeit made in a different context, in Ref. 41.

(g) It is argued in Ref. 32 that the FCC lattice is more typical for the internal structure of real proteins than the SC lattice. Indeed, it seems that it leads more easily to a hydrophobic core construction, otherwise it would be hard to understand why the code of Ref. 32 is so much more efficient than that of Ref. 30, 31. It finds for most randomly chosen HP sequences of length $N = 100$ the ground state(s), while the SC code of Ref. 30, 31 in general breaks down at $N \approx 80$. The same is true for nPERM. We found that we could reach the true ground states for several randomly chosen sequences of length $N = 80$ on the FCC lattice[42], while chains of the same length on the SC lattice would have posed serious problems.

3.2 Other Models

In Ref. 22 we had used (old) PERM to find the ground state of a chain with two types of monomers which were however not of the H-P type[43]. This model is interesting because it shows an unexpected transition from a 4-helix bundle to a β-sheet dominated structure at very low temperatures. Later the same ground state was found also in Ref. 9 using a different Monte Carlo method. With nPERM the ground state is found an order of magnitude faster.

Models with more than 2 types of monomers were studied also in Ref. 22, with no particular difficulties. Applying nPERM to the model of Ref. 29 which has 5 types of monomers gave very fast convergence (which should not surprise in view of the very small length of this chain), and also to a very stable estimate of excited states. Up to the tenth excited energy level the degeneracy factors were so close to integers that one could infer the exact numbers of excited states, although they went up to the hundreds (unpublished). The fact that PERM and nPERM are truly thermodynamic approaches and give thus rise to the full energy spectrum is indeed one of their strong points.

4 Off-Lattice Models

There are much less results for off-lattice models in the literature that can be used as benchmarks. Of course one could use realistic small proteins with one of the standard force fields, but the results would be hard to compare with other algorithms. Thus we preferred simple toy models. There is essentially one such toy model in the literature, by Stillinger *et al.*[44]. This is a two-dimensional model with modified Lennard-Jones potentials, fixed bond lengths, and a weak bond angle stiffness term. It contains also two types of monomers mimicking hydrophobic and polar amino acids, but this time they are called "A" (hydrophobic) and "B" (polar). In Ref. 44 the authors studied only Fibonacci sequences where the chain length is a Fibonacci number F_i, and there are F_{i-2} A monomers and F_{i-1} B's.

Putative ground states were found in Ref. 25 by first using nPERM to come close to a potential minimum, and then using conjugate gradient descent to reach the minimum itself. Results are depicted in Fig. 3. In all cases the energies are smaller than those reported in Ref. 44, e.g. for $N = 55$ the configuration shown in Fig. 3 has $E = -18.515$, while the supposed ground state given in Ref. 44 had $E = -14.41$. For the longer chains

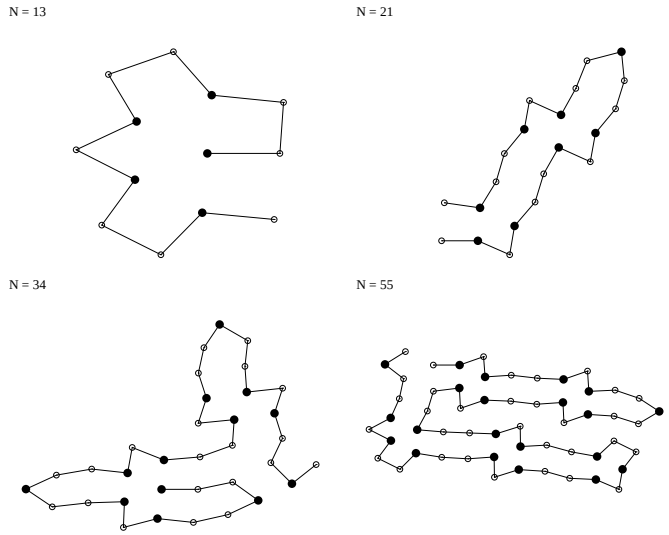

Figure 3. Putative 2-d ground states of Fibonacci sequences. Full dots indicate hydrophobic monomers.

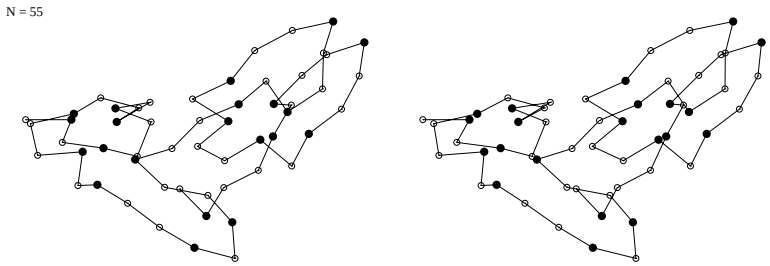

Figure 4. Stereographic view of the putative ground state of the $3-d$ Fibonacci sequence with $N = 55$. Again, A monomers are shown as filled circles.

also the topology is different. A very striking feature is that only the shortest chain (with $N = 13$) has a connected hydrophobic core. Thus this is a very bad model for real proteins. The ultimate reason for this is the absence of longer connected stretches of hydrophobic monomers along the chain. In a Fibonacci chain as defined in Ref. 44, all A monomers are flanked by two B monomers, which prevents the formation of large hydrophobic cores.

The same model was used in Ref. 25 for 3-d structures. The potential energy was kept identical to the above, although a different bond angle term favouring a more Ramachandran-like torsion angle pattern would have been more physical. But the Fibonacci structure of the AB sequences prevented also in 3-d the formation of a single core (see Fig. 3), and thus the model is not very realistic anyhow. We present it here merely for future benchmarks.

5 Conclusion

In this review we have not attempted to study very realistic protein models. Rather, we wanted to present a class of general purpose algorithms which perform extremely well for simple lattice polymers, and which therefore have a potential of being useful also in more complex applications. When applied to lattice models, nPERM seems to be superior than genetic algorithms and all known Markov chain (Metropolis-type) Monte Carlo methods, and is matched only by exact methods which are however very specifically taylored to these models. For off-lattice models the same algorithm fares quite well, even if it might not be better than modern Markov chain methods. We have not yet tested it for proteins surrounded by solvents, where previous Monte Carlo methods seem (still) inferior to molecular dynamics. As it is formulated now, we expect that PERM and nPERM would share the same problems as Metropolis-type Monte Carlo. But we cannot exclude that future progress might lead to substantial improvements. We just want to mention that a first step has been made in Ref. 45, 46, where PERM was combined with the multicanonical (umbrella sampling) concept, leading to very promising results. In the long run, we expect that a mix of different Monte Carlo strategies will lead finally to a substantially improved understanding of protein folding.

Acknowledgements

I want to thank my collaborators U. Bastolla, H. Frauenkron, E. Gerstner, S. M. Causo, B. Coluzzi, W. Nadler, H.-P. Hsu, and V. Mehra for the very fruitful and pleasant joint efforts. I also want to thank B. Berg and U. Hansmann for many useful discussions.

References

1. L. M. Gierasch and J. King (eds.), *Protein Folding, Deciphering the Second Half of the Genetic Code* (AAAS, New York, 1990).
2. T.E. Creighton (ed.), *Protein Folding* (Freeman, New York, 1992).
3. K. M. Merz Jr. and S. M. LeGrand (eds.), *The Protein Folding Problem and Tertiary Structure Prediction* (Birkhäuser, Boston, 1994).
4. H. Bohr and S. Brunak(eds.), *Protein Folds: A Distance Based Approach* (CRC Press, Boca-Raton/FL. 1996).
5. Y. Okamoto, e-print cond-mat/0308360 (2003).
6. T. Wille and U.H.E. Hansmann, Phys. Rev. Lett. **88**, 068105 (2002).
7. F. Wang and D.P. Landau, Phys. Rev. E **64**, 056101 (2001).
8. A. Irbäck, "Dynamical-parameter algorithms for protein folding", in *Monte Carlo Approach to Biopolymers and Protein Folding*, eds. P. Grassberger *et al.*, pp. 98-109 (World Scientific, Singapore, 1998).
9. G. Chikenji, M. Kikuchi, and Y. Iba, Phys. Rev. Lett. **83**, 1886 (1999).
10. B. Berg and H.-P. Hsu, e-print cond-mat/0306589 (2003).
11. H.H. Gan, A. Tropsha, and T. Schlick, J. Chem. Phys. **113**, 5511 (2000).
12. H. P. Schwefel, *Evolution and Optimum Seeking* (Wiley, New York, 1995).
13. C.J. Umrigar, M.P. Nightingale, and K.J. Runge, J. Chem. Phys. **99**, 2865 (1993).
14. H. Kahn, *'Use of Different Monte Carlo Sampling Techniques'*, in ed. H.A. Meyer, *Symposion on the Monte Carlo Method* (Wiley, New York 1956).

15. J.S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Series in Statistics (Springer, New York 2001).
16. A. Doucet, Nando de Freitas, and Neil Gordon, *Sequential Monte Carlo Methods in Practice* (Springer, New York, 2001).
17. B. Velikson, T. Garel, J.-C. Niel, H. Orland and J.C. Smith, J. Comput. Chem. **13**, 1216 (1992).
18. J.L. Zhang and J.S. Liu, J. Chem. Phys. **117**, 3492 (2002).
19. R. Tarjan, SIAM J. Comput. **1**, 146 (1972).
20. H.-P. Hsu, V. Mehra, W. Nadler, and P. Grassberger, J. Chem. Phys. **118**, 444 (2003).
21. H. Frauenkron, U. Bastolla, E. Gerstner, P. Grassberger, and W. Nadler, Phys. Rev. Lett. **80**, 3149 (1998).
22. U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler, PROTEINS: Structure, Function, and Genetics **32**, 52 (1998).
23. H. Frauenkron, U. Bastolla, E. Gerstner, P. Grassberger, and W. Nadler, in 'Monte Carlo Approach to Biopolymers and Protein Folding', eds. P. Grassberger *et al.* (World Scientific, 1998).
24. P. Grassberger, Phys. Rev. **E** 56, 3682 (1997).
25. H.-P. Hsu, V. Mehra, and P. Grassberger, e-print cond-mat/0302545 (2003).
26. K.A. Dill, Biochemistry **24**, 1501 (1985).
27. K.F. Lau and K.A. Dill, Macromolecules **22**, 3986 (1989).
28. H.S. Chan and K.A. Dill, J. Chem. Phys. **95**, 3775 (1991).
29. H. Kaya and H. S. Chan, Phys. Rev. Lett. **85**, 4823 (2000).
30. K. Yue and K.A. Dill, Phys. Rev. E **48**, 2267 (1993).
31. K. Yue and K.A. Dill, Proc. Natl. Acad. Sci. USA **92**, 146 (1995).
32. R. Backofen and S. Will, *A Constraint-Based Approach to Structure Prediction for Simplified Protein Models that Outperforms Other Existing Methods.*, in Proc. XIX. Internat. Conf. on Logic Programming (ICLP 2003).
33. K. Yue *et al.*, Proc. Natl. Acad. Sci. USA **92**, 325 (1995).
34. R. König and T. Dandekar, BioSystems **50**, 17 (1999).
35. F. Liang and W. H. Wong, J. Chem Phys. **115**, 3374 (2001).
36. K.A. Dill, K. Fiebig, and H.S. Chan, Proc. Natl. Acad. Sci. USA **90**, 1942 (1993).
37. E.E. Lattman, K.M. Fiebig, and K.A. Dill, Biochenistry **33**, 6158 (1994).
38. L. Toma and Toma, Protein Sci. **5**, 147 (1996).
39. R. Unger and J. Moult, *A genetic algorithm for 3d protein folding simulations*, in 5th Proc. Int'l. Conf. on Genetic Algorithms, pp. 581-588 (1993).
40. A.L. Patton, W.F. Punch III, and E.D. Goodman, *A standard GA approach to native protein conformation prediction*, in 6th Proc. Int'l. Conf. on Genetic Algorithms, pp. 574-581 (1995).
41. R. König and T. Dandekar, Protein Eng. **14**, 329 (2001).
42. For a publicly accessible server which folds arbitrary HP sequences of length up to 100, see the URL `http://www.bio.inf.uni-jena.de/Software/Prediction/prediction.cgi`.
43. E. M. O'Toole and A. Z. Panagiotopoulos, J. Chem. Phys **97**, 8644 (1992).
44. F. H. Stillinger and T. Head-Gordon, Phys. Rev. E **52**, 2872 (1995).
45. M. Bachmann and W. Janke, e-print cond-mat/0304613 (2003).
46. T. Prellberg, private communication (2003).