A Branch and Bound Algorithm for the Protein Folding Problem in the HP Lattice Model

Mao Chen* and Wen-Qi Huang

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China.

A branch and bound algorithm is proposed for the two-dimensional protein folding problem in the HP lattice model. In this algorithm, the benefit of each possible location of hydrophobic monomers is evaluated and only promising nodes are kept for further branching at each level. The proposed algorithm is compared with other well-known methods for 10 benchmark sequences with lengths ranging from 20 to 100 monomers. The results indicate that our method is a very efficient and promising tool for the protein folding problem.

Key words: protein folding, HP model, branch and bound, lattice

Introduction

The protein folding problem, or the protein structure prediction problem, is one of the most interesting problems in biological science. Studies have indicated that proteins' biological functions are determined by their dimensional folding structures. Because the structure of a protein is strongly correlated with the sequence of amino acid residues, predicting the native conformation of a protein from its given sequence is a feasible approach and is of great significance for the protein engineering. Since the problem is too difficult to be approached with fully realistic potentials, the theoretical community has introduced and examined several highly simplified models. One of them is the HP model of Dill *et al* (*1–3*) where each amino acid is treated as a point particle on a regular (quadratic or cubic) lattice, and only two types of amino acids—hydrophobic (H) and polar (P)—are considered.

Although the HP model is extremely simple, it still captures the essence of the important components of the protein folding problem (*4*). The protein folding problem in the HP model has been shown to be NP-complete, and hence unlikely to be solvable in polynomial time (*5–7*). For relatively short chains, an exact enumeration of all the conformations is possible. In dealing with longer chains, however, more efficient approximation algorithms are certainly desirable.

The methods used to find low energy structures of the HP model include genetic algorithm (GA; ref. *8–12*), Monte Carlo (MC; ref. *10*, *12*), simulated annealing (*9*), *etc.* These algorithms can find optimal or near-optimal energy structures for most benchmark sequences, however, their computation time is rather long. In this paper, a branch and bound algorithm is proposed to find the native conformation for the two-dimensional (2D) HP model. The experimental results have shown that our algorithm is very efficient, which can find optimal or near-optimal conformations in a very short time for a number of sequences with lengths ranging from 20 to 100 monomers.

*** Corresponding author.**
E-mail: mchen_1@163.com

Model

Let us consider this problem in 2D Euclidean space. The monomers are numbered consecutively from 1 to n along the chain, which is folded on the square lattice, and each monomer occupies one site with the center on the lattice point. Note that each monomer should be connected to its chain neighbors and is unable to occupy a site filled by other monomers. If monomer i is placed on the square lattice, then the coordinates of its location are denoted by (x_i, y_i).

The HP model is based on the assumption that the hydrophobic interaction is one of the fundamental principles in the protein folding. An attractive hydrophobic interaction provides for the main driving force for the formation of a hydrophobic core that is screened from the aqueous environment by a shell of polar monomers. Therefore, the energy function of the HP model is defined as:

$$E = -\sum_{i,j<i-1} \sigma_i \, \sigma_j \qquad (1)$$

where $\sigma_i = 1$ if the i^{th} monomer in the chain is hydrophobic, otherwise $\sigma_i = 0$. In other words, the energy of a conformation can be obtained by counting the number of adjacent pairs of hydrophobic monomers (H–H) that are not consecutively numbered, and multiplying by -1. The goal of the protein folding problem is to find the conformation with the minimal energy.

Figure 1 shows a folding conformation of sequence HPPHPPHPHPPHP on the 2D square lattice. It can be seen that each monomer occupies one lattice site connected to its chain neighbors. The energy of this conformation is -4, which is the lowest energy state of the sequence. Obviously, there is a compact hydrophobic core in the folded conformation.

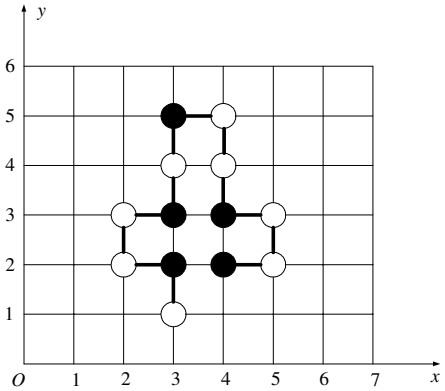

Fig. 1 The lowest energy conformation with E = −4 of sequence HPPHPPHPHPPHP. Black point particle: hydrophobic (H); White point particle: polar (P).

Algorithm

In our algorithm, a conformation is built by adding a new monomer at an allowed neighbor site of the last placed monomer on the square lattice. In order to obtain a self-avoiding conformation, an already occupied neighbor should not be considered. The monomers are placed consecutively until all the n (the length of the chain) monomers are placed, that is, our algorithm is a growth algorithm.

If $k-1$ $(1 \leq k \leq n)$ monomers have been placed on the square lattice, the k^{th} monomer may have three possible locations: turn 90° right, turn 90° left, or continue ahead. Figure 2 gives a partial conformation where four monomers have been placed on the square lattice. It can be seen that there are three unoccupied positions neighboring to Monomer 4. The next monomer, namely Monomer 5, can be placed at any one of these unoccupied positions, resulting in three different partial conformations accordingly. In this way, all possible folding conformations of a sequence can be enumerated. As shown in Figure 3, a search tree representation can be used to denote all possible folding conformations, with three descendants at most for each node. Each node in the search tree corresponds to a partial conformation, and a line between two nodes represents a placement choice of a new monomer to the existing partial conformation. Consequently, leaf nodes at the end of the tree correspond to the complete conformation.

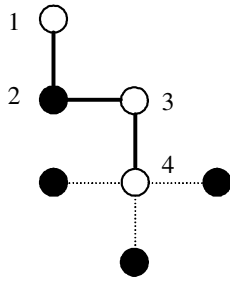

Fig. 2 The three possible positions for Monomer 5.

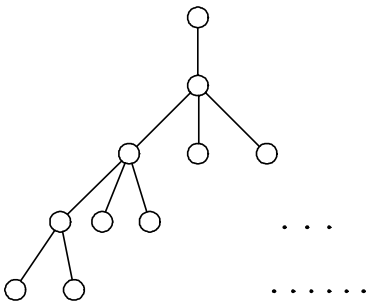

Fig. 3 A representation of the search tree.

From Figure 3, it is obvious that the conformational space grows exponentially when the length of the protein chain increases. As mentioned by Unger and Moult (*12*), the number of possible (self-avoiding) conformations for an L-long sequence on a 2D square lattice is $A\mu^L L^\gamma$, where $\mu \approx 2.63$ and $\gamma \approx 0.333$. Accordingly, for a protein chain of not too short length, the search space is too huge to find the lowest energy conformation within a reasonable running time.

To reduce the computational cost, a so-called branch and bound method is introduced in this paper. In this search method, only the promising nodes are kept for further branching and the remaining nodes are pruned off permanently. Since a large part of the search tree is pruned off aggressively to obtain a solution, its running time is polynomial in the size of the problems.

In our algorithm, we treat H monomers and P monomers differently. For a partial conformation where $k-1$ monomers have been placed on the square lattice, if the k^{th} monomer is P, then all possible branches should be kept. Otherwise, if the k^{th} monomer is H, then the benefit of all possible branches of the k^{th} monomer will be evaluated and some branches may be pruned. That is to say, the main part of our algorithm is centered on the evaluation and pruning of the H monomers. This strategy maintains the diversity of the conformations and eliminates the hopeless partial conformation at the same time. The details are as follows:

We set two variables, U_k and Z_k, as the thresholds to evaluate the benefit of all branches for monomer k. Here, U_k is defined as the lowest energy of the partial conformation with length k that has ever been generated so far, and Z_k is the arithmetic average energy of the partial conformation with length k so far. After pseudo-placing monomer k at a possible location, we calculate E_k, which is defined as the energy of the current partial conformation with k monomers placed. It should be pointed out that the term "pseudo-place" means that it is just a test and the placing process can be reverted. Then we compare E_k with thresholds U_k and Z_k:

If $E_k \leq U_k$, it means that this partial conformation is very promising and this branch should be kept. If $E_k > Z_k$, that means the benefit of the partial conformation is below the average, so this conformation is discarded with probability ρ_1. Otherwise, if $Z_k \geq E_k > U_k$, the partial conformation is discarded with probability ρ_2.

The pseudo-code of this subroutine is presented in Figure 4, including the details of evaluation criterion and the pruning mechanism, which is the main part of our algorithm.

```
Procedure: Searching (E_{k-1}, k)
Begin
    Compute M_k as the set of possible sites for monomer k
    If |M_k |>0
        For each candidate site α∈ M_k, do
            Calculate E_k of the partial conformation after pseudo-placing monomer k at α;
            If k=n                                          /* the conformation hit n */
                Place monomer k at α and update E_min by E_n;
                Return;
            Else
                If monomer k is H (hydrophobic)
                    If E_k ≤ U_k                            /* all branches are kept */
                        Place monomer k at α;
                        Call Searching (E_k, k+1);
                    If E_k>Z_k                              /* prune with probability ρ_1 */
                        Draw r uniformly ∈ [0,1]
                        If  r > ρ_1
                            Place monomer k at α;
                            Call Searching (E_k, k+1);
                    If E_k ∈ [U_k , Z_k]                    /* prune with probability ρ_2 */
                        Draw r uniformly ∈ [0,1]
                        If  r > ρ_2
                            Place monomer k at α;
                            Call Searching (E_k, k+1);
                Else                                        /* the k^th monomer is polar */
                    Place monomer k at α;
                    Call Searching (E_k, k+1);
End.
```

Fig. 4 The pseudo-code of the subroutine in the branch and bound algorithm.

The above process is implemented in a recursive way until all the conformations are either pruned or hit length n. From the conformations hitting length n, we choose one with the lowest energy as the output of the algorithm. It should be mentioned that the search could be implemented by depth-first or breadth-first, where the two results are identical. In this paper, our algorithm is implemented by depth-first.

Here, E_{min} is the minimal energy of the complete conformations ever built. Note that the first two monomers of a chain can be placed on the square lattice randomly. Therefore, the input parameters are $k = 3$, $E_2 = 0$. The initial values of the two thresholds U_k and Z_k are both 0.

Obviously, if $\rho_1 = 0$ and $\rho_2 = 0$, the search space will be the complete tree (no node be pruned) and it will take a prohibitively long time to search for the lowest energy conformation. If $\rho_1 = 1$ and $\rho_2 = 1$, it takes a very little time to search the entire search space because the thresholds are so high that many promising nodes may be discarded. That is to say, the higher the value of the probabilities, the more difficult a branch is to be kept. Therefore, choosing the value of ρ_1 and ρ_2 is an essential factor affecting the speed and efficiency of this approach. In this paper, we let $\rho_1 = 0.8$ and $\rho_2 = 0.5$. The probability ρ_2 is chosen to be less than ρ_1 because a partial conformation with energy below average is more promising than a high energy partial conformation.

In this way, E_k, the energy of the partial conformation, can be viewed as the energy expectation of the partial conformation after looking one step ahead and Z_k is expressed as the mean energy of the already generated partial conformations of length k. Z_k keeps a historical record, which is, to a large extent, conducive to the formulation of promising conformations. For any partial conformation, it would have more opportunities to procreate if holding higher individual quality (E_k), which is in accordance with the law of natural selection.

Validation

To test the performance of the branch and bound algorithm, we compared it with the MC, GA, and mixed search (MS; ref. *13*) algorithms by using 10 benchmark sequences for evaluation (Table 1).

Table 2 presents the results obtained by the four methods on the 10 different sequences. As shown in the table, our branch and bound algorithm can find the optimal lowest energy conformations for six sequences. It is noteworthy that our algorithm can find one native state for the sequence of length 60, whereas the other three methods failed. For the two long sequences of length 85 and 100, respectively, our algorithm can find near-optimal energy conformations. It should be pointed out that predicting the longest sequence of length 100 is a hard problem, whose native state can only be obtained by a few methods such as the PERM algorithm (*14*, *15*) and the guided simulated annealing method (*7*).

Table 1 The 10 Benchmark Sequences for Algorithm Evaluation

Length	Sequence
20	HPHPPHHPHPPHPHHPPHPH
24	HHPPHPPHPPHPPHPPHPPHPPHH
25	PPHPPHHPPPPHHPPPPHHPPPPHH
36	PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP
48	PPHPPHHPPHHPPPPPHHHHHHHHHHPPPPPHHPPHHPPHPPHHHHH
50	PPHPPHPHPHHHHPHPPPHPPPHPPPPHPPPHPPPHPHHHHPHPHPHPHH
60	PPHHHPHHHHHHHHPPPHHHHHHHHHHPHPPPHHHHHHHHHHPPPPHH– HHHHPHHPHP
64	HHHHHHHHHHHHPHPHPPHHPPHHPPHPPHHPPHHPPHPPHHPPHPPHP– HPHHHHHHHHHHHH
85	HHHHPPPPHHHHHHHHHHHHPPPPPPHHHHHHHHHHHHPPPHHHHHHHHH– HHHPPPHHHHHHHHHHHHPPPHPPHHPPHHPPHPH
100	PPPHHPPHHHHPPHHHPHHPHHPHHHHPPPPPPPHHHHHHPPHHHHHHP– PPPPPPPPHPHHPHHHHHHHHHHHPPHHHPHHPHPPHPHHHPPPPPPHHH

Table 2 Performance Comparison of the Four Algorithms*

Length	Optimal	MC	GA	MS	BB
20	−9	−9	−9	−9	−9
24	−9	−9	−9	−9	−9
25	−8	−7	−8	−8	−8
36	−14	−12	−14	−14	−14
48	−23	−18	−22	−22	−22
50	−21	−19	−21	−21	−21
60	−36	−31	−34	−34	−36
64	−42	−31	−37	−38	−38
85	−53	N/A	N/A	N/A	−52
100	−50	N/A	N/A	N/A	−48

*Performance comparison on finding the lowest energy conformations of the four algorithms, including Monte Carlo (MC), genetic algorithm (GA), mixed search (MS), and branch and bound (BB).

We did not compare the speed with other methods directly because the machines were different. Moreover, the running time of the other three methods was presented in terms of "number of steps" while the exact CPU time was used in our test. All the computations in this study were carried on a 2.4 GHz PC with 512 M memory. The CPU time for all sequences was less than 10 s except the sequence of length 64, for which the CPU time was 39.46 s. It can be seen from Unger and Moult (*12*) that the "number of steps" of MC and GA methods increases badly with the increase of sequence lengths, therefore, it is imaginable that the computational speed of MC and GA methods in Unger and Moult (*12*) for practical applications is unacceptable.

The resulting folding conformations for sequences with 24, 36, 60, 85, and 100 monomers are given in Figure 5, respectively. For sequences with 24, 36, and 60 monomers, the corresponding conformations are all of the lowest energy. For the other two sequences with longer lengths, the corresponding conformations are also of near-optimal energy. It can be seen that the conformation has a single compact hydrophobic core for all sequences, which is analogous to the real protein structure.

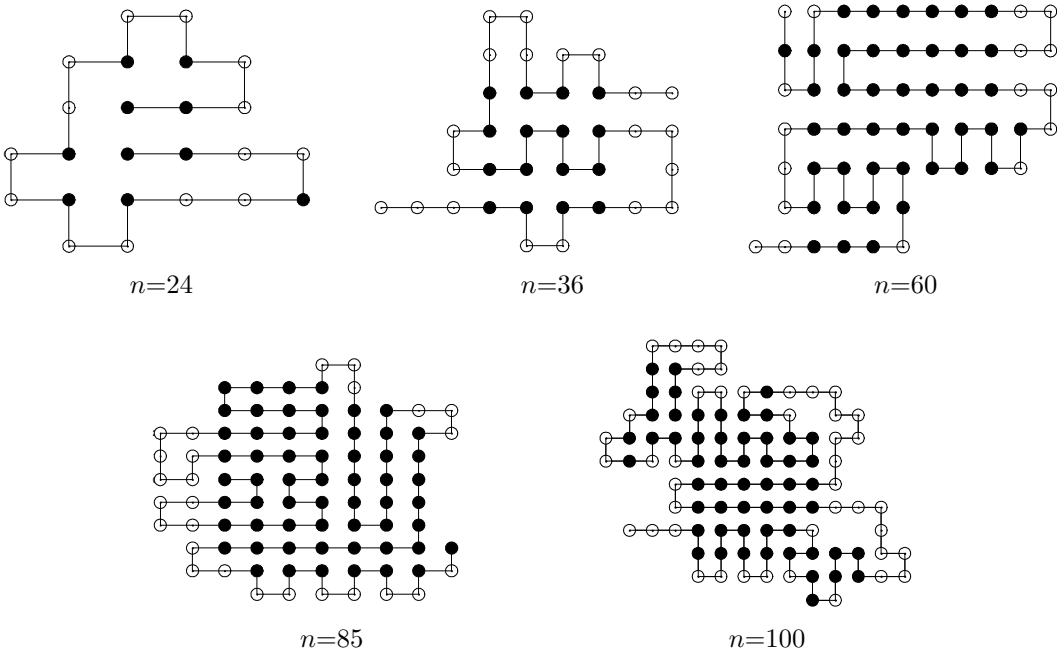

Fig. 5 The lowest energy states of the sequences with length n = 24, 36, 60, 85, and 100, respectively.

Conclusion

The branch and bound algorithm proposed in this paper is a novel and effective tool for the conformational search in the low-energy regions of the protein folding problem in the 2D HP model. The experimental results on 10 benchmark sequences demonstrate that our algorithm outperforms other three methods in terms of speed and efficiency. Our algorithm is similar to the "population control" scheme (*15*) where individuals would have more opportunities to procreate if holding higher individual quality, and the pruning mechanism reduces considerably the computational burden of search. This is the root reason why our approach yields high efficiency.

With slight modification, this algorithm can be extended for the 3D version. We should point out that, the coding of this algorithm is very simple and hence it can be easily implemented by practitioners.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 10471051) and the National Basic Research Program (973 Program) of China (No. 2004CB318000).

References

1. Dill, K.A. 1985. Theory for the folding and stability of globular proteins. *Biochemistry* 24: 1501-1509.
2. Dill, K.A., *et al.* 1995. Principles of protein folding: a perspective from simple exact models. *Protein Sci.* 4: 561-602.
3. Dill, K.A., *et al.* 1993. Cooperativity in protein-folding kinetics. *Proc. Natl. Acad. Sci. USA* 90: 1942-1946.
4. Lau, K.F. and Dill, K.A. 1990. Theory for protein mutability and biogenesis. *Proc. Natl. Acad. Sci. USA* 87: 638-642.
5. Berger, B. and Leighton, T. 1998. Protein folding in the hydrophilic-hydrophobic (HP) model is NP-complete. *J. Comput. Biol.* 5: 27-40.
6. Crescenzi, P., *et al.* 1998. On the complexity of protein folding. *J. Comput. Biol.* 5: 423-465.
7. Hart, W.E. and Istrail, S. 1997. Robust proofs of NP-hardness for protein folding: general lattices and energy potentials. *J. Comput. Biol.* 4: 1-22.
8. Konig, R. and Dandekar, T. 1999. Improving genetic algorithms for protein folding simulations by systematic crossover. *Biosystems* 50: 17-25.
9. Chou, C.I., *et al.* 2003. Guided simulated annealing method for optimization problems. *Phys. Rev. E* 67: 066704.
10. Metropolis, N., *et al.* 1953. Equation of state calculations by fast computing machine. *J. Chem. Phys.* 21: 1087-1092.
11. Sun, S. 1993. Reduced representation model of protein structure prediction: statistical potential and genetic algorithms. *Protein Sci.* 2: 762-785.
12. Unger, R. and Moult, J. 1993. Genetic algorithms for protein folding simulations. *J. Mol. Biol.* 231: 75-81.
13. Huang, J., *et al.* 2003. Mixed search algorithm for protein folding. *Wuhan Univ. J. Nat. Sci.* 8: 765-768.
14. Hsu, H.P., *et al.* 2003. Growth algorithms for lattice heteropolymers at low temperatures. *J. Chem. Phys.* 118: 444-451.
15. Huang, W. and Lü, Z. 2004. Personification algorithm for protein folding problem: improvements in PERM. *Chin. Sci. Bull.* 49: 2092-2096.