An Efficient Algorithm for Computing the Fitness Function of a Hydrophobic-Hydrophilic Model

Md. Tamjidul Hoque, Madhu Chetty and Laurence S Dooley
Gippsland School of Computing and Information Technology
Monash University, Churchill VIC 3842, Australia
E-mail: {Tamjidul.Hoque, Madhu.Chetty, Laurence.Dooley}@infotech.monash.edu.au

Abstract

The protein folding problem is a minimization problem in which the energy function is often regarded as the fitness function. There are several models for protein folding prediction including the Hydrophobic-Hydrophilic (HP) model. Though this model is an elementary one, it is widely used as a test-bed for faster execution of new algorithms. Fitness computation is one of the major computational parts of the HP model. This paper proposes an efficient search (ES) approach for computing the fitness value requiring only O(n) complexity in contrast to the full search (FS) approach that requires $O(n^2)$ complexity. The efficiency of the proposed ES approach results due to its utilization of some inherent properties of the HP model. The ES approach represents residues in a cartesian coordinate framework and then uses relative distance and coordinate polarity to reduce complexity.

Keywords: HP model, fitness function, improved computation, relative distance, relative polarity.

1. Introduction

There are 20 different amino acids that lead to the formation of different proteins. Thus, each protein is essentially a sequence of one-dimensional chain of amino acids that adopts a specific folded three dimensional shape called its native conformation. Each shape provides valuable clues to the protein function. One reason for the significant interest in protein folding problem in general and efficient techniques in particular is the large amount of genetic information produced by the Human Genome Project. Because of the importance and computationally intensive nature of this problem, it has been identified as a National Grand Challenge in bio-chemistry in the United States (Lamont). Not only are the research efforts directed towards determining the in-vivo structures of naturally occurring proteins but efforts are also going on for promoting faster protein design than is currently possible. The latter is referred to as the 'inverse protein folding problem'.

It is widely believed [10] that the native conformation of a protein is determined by the amino acid sequence of the protein under several regular forces. Amino acids can be categorized as either positively or negatively charged, side chain size as tiny, small or large, aliphatic or aromatic etc. One of the property that affects folding strongly is known as *hydrophobicity*, based on which the amino acid residues can be divided into two groups. Hydrophobic (H) or non-polar residues stay away from water (*water-hating*) [1], and tend to be inside the protein core while the hydrophilic or polar (P) residues are attracted towards water (*water-loving*), and hence tend to remain outside the protein core.

Based on this property, the HP model was introduced [3] that has been used by many researchers for protein folding prediction. A native conformation for a string of amino acids is the one which has the lowest energy and is achieved when the numbers of hydrophobic-hydrophobic (H-H) pairs, called topological neighbour (TN), is maximized. The TN, by definition, are those adjacent H pairs that have unit lattice distance but are not sequential or are not connected. Further, in a HP-model, the fitness function is most frequently computed value for protein structure prediction. In most computational search schemes, such as those involving genetic algorithms (GA), the fitness is computed in every iteration, for each chromosome for the selection of a better seed [6][11][12]. Thus, faster and quicker computation of these functions will make the search process fast and efficient. Moreover, as protein folding in the HP model is NP-complete [2], any improvement in the fitness computation will also have a corresponding impact on

Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)
0-7695-2291-2/04 $ 20.00 IEEE

the overall performance of the GA as well. Several approximation algorithms have also been approached [5][8][9][10].

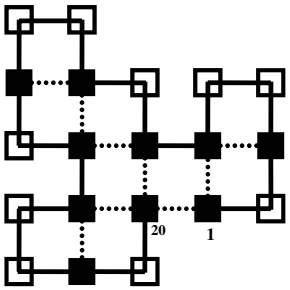
This paper proposes an efficient search algorithm for protein folding problem. We study this problem of improving the computation of the fitness function in a two-dimensional (2D) square lattice. The remainder of the paper is organized as follows. In Sec. 2, the HP model and formulation of amino acid chain string is described together with the full search (FS) approach to compute the fitness value. In Sec. 3, Lemmas which help to improve the FS are proposed and issues related efficient search (ES) strategies are discussed. In Sec. 4, computational complexity analysis of ES is presented. Finally, conclusions are given in Sec. 5.

2. HP model

For any given amino acid sequence, different orientations are possible. As mentioned earlier, the higher the number of TN pairs (i.e. by definition, the lower the fitness function) for any orientation, the closer the confirmation will be to the desired folding.

2.1. HP model and fitness function

In a 2D HP model, the orientation is represented by placing the amino acid chain on a square lattice model and then forming a self-avoiding walk. Figure 1 shows an HP model where the hydrophobic and hydrophilic residues are represented as black and white squares respectively. A solid line connecting the squares indicates the sequence of the amino acids chain while the dotted line indicates TN pairs.



■ Hydrophobic residue, □ Hydrophilic residue
Fitness Value =-9

Figure 1. An HP model comprising of hydrophobic and hydrophilic residues.

The following approach, given in [4], has been used in the proposed work for measuring the fitness function.

1) Initialize fitness function F=0
2) Compute and identify all possible pairs of TN in the HP model
3) For each of these pairs, decrement fitness function, F

To compute F, the string S is traversed to determine the number of TN pairs in the HP model. It can be seen that in Figure 1, the number of such pairs is 9 and the value of fitness function is therefore equals to -9.

In a 2D placement, the residues of the string can be represented by their cartesian coordinates (x, y). For the sake of simplicity and without loss of generality, it is assumed that the starting hydrophobic residue 1 is at (0, 0) [see Table 1].

2.2. Binary string formulation

For computational ease, we can represent the hydrophobic and hydrophilic residues as binary '1' and binary '0' respectively. Thus, the chain of amino acid sequence given in Figure 1 can be represented as $S = [10100110100101100101]$. A binary '1' at an odd index is called an *odd-1* and at an even index is called *even-1* [9].

Table 1. Coordinates and relative lattice distance

Row 1: Hydrophobic residue index with reference to Figure 1.

	1	3	6	7	9	12	14	15	18	20
x	0	1	0	-1	-2	-3	-2	-2	-2	-1
y	0	1	1	1	2	2	1	0	-1	0
		2	1	2	4	5	3	2	3	1

Row 4: Relative lattice distance respect to residue at (0, 0).

For any string, there is a fixed range of values of the fitness function, F given as 0, -1, -2, …, -M. The maximum value M is given as

$$M = 2 * \min(E[S], O[S]) \quad (1)$$

where, O[S] = number of *odd-1* in the string
E[S] = numbers of *even-1* in the string

It is assumed that neither of the end points in the string are hydrophobic residues. It is obvious that, on a square lattice, an even-1 will always be adjacent to odd-1. Hence, each element in the string S (except the

two end points) can have a maximum of two topological neighbours.

The string $S \in \{0, 1\}^m$ can be represented as binary string, $S = [s_1, s_2, s_s, \cdots, s_m]$. Let us assume S′ to be the string having n hydrophobic residues only, i.e. $S' = [s'_1, s'_2, s'_3, \cdots, s'_n]$. The string S′ will be a subset of S. The relative lattice distance, d_i is measured from s'_1 to any s'_i, $d_i = |x_i| + |y_i|, where \quad (2 \le i \le n)$. The values of d for various residues are shown in the last row of Table 1. Note that only non-diagonal lattice distance is assumed so that distance computation is always integer based. Further, the index of the hydrophobic residue in string S′ is the same as the index given in the original string S. Thus, index of a residue, say 3, remains 3 in both S′ and S. These index values are shown in the top row of Table 1.

3. Full search (FS) approach

In the full search (FS) approach to compute the fitness function F, s'_1 is first compared with s'_2, s'_3 … s'_n; then s'_2 is compared with s'_3, s'_4 …, s'_n, and so on. During comparison between (s'_i, s'_j), where $i \neq j$, if the (non-diagonal) distance = the unit lattice, then F is decremented (F=F-1). The initial value of F is assumed as F=0. The complete steps involved in the FS approach are given in Algorithm 1 below.

Algorithm 1. Full search approach

Precondition:
Fitness function F=0; S′ (= s'_1, s'_2, s'_3 …, s'_n);
Coordinates of the hydrophobic residue of S′;

Post condition: Fitness value F.

```
1.  FOR  i (1: n-1) DO
2.      FOR j (i+1 : n) DO
3.          Compute distance d between (s'i, s'j)
4.          IF |d| = 1 then decrement F
5.      ENDFOR
6.  ENDFOR
```

Hence, if the total number of hydrophobic residues is *n*, then with FS approach, the computation of F requires a time complexity of $\boldsymbol{O}(n^2)$.

4. Improving FS time complexity

By considering the even and odd index positions of a ‘1’ in the string S, ‘1’ can be grouped as either *even-1* and *odd-1* according to whether the index number. The orientation of the members of these groups elicits some useful properties which can be exploited to reduce time complexity.

4.1. Toward an efficient search approach

The following series of lemma are presented which form the basis for constructing an efficient search (ES) approach.

Lemma 1: For any particular lattice point, the relative lattice distance of any *odd-1* and any *even-1* will never be the same.

Proof: In a square lattice, an *odd-1* can only be adjacent to *even-1* and vice versa. Adjacent odd-1s differ from an even-1 by minimum of one lattice distance. So, the distance of *even-1* in a particular lattice and the adjacent *odd-1* with respect to that lattice point will always differ by an odd number by induction. □

Lemma 2: From any lattice point, if the relative lattice distance for any *odd-1* is even then, all the *odd-1* will have even lattice distance and all *even-1s* will have an odd lattice distance with respect to that point.

Proof: Using Lemma 1, the distance from a particular lattice point to any *odd-1* and from that particular lattice point to any *even-1* differ by odd number. Thus if the distance of particular point from an *odd-1* is odd then the distance of any *even-1* from that particular point is even and visa versa. By induction, this extends to all *odd-1*s and *even-1*s. □

Lemma 3: The relative distance between any two *odd-1*s and also between any two *even-1*s is always even.

Proof: Using Lemma 2, from any particular point if any *odd-1* has an even distance, then all *odd-1*s will have an even distance. The same is also true for any two *even-1s*. □

To calculate the relative distance (i.e. last row of Table 1) for all hydrophobic residues with respect to a particular hydrophobic residue (i.e. s'_1), the following conditions are given.

a) Various subsets (called equidistant subsets) are formulated comprising of residues which are equidistant from the reference residue (i.e. s'_1). Using Lemma 1, *odd-1*s and *even-1*s fall into different equidistant subset.
b) Using Lemma2, if a particular point is *odd-1* then all *even-1*s will be odd distance from that point and all *odd-1*s will be even distance.
c) Using Lemma 3, for any hydrophobic residue, some *odd-1s* and some *even-1s* are alternatively separated on the basis of relative distance.

d) To compute F, *odd-1*s are only compared with the next adjacent *even-1* (if it exists) which are separated by unit relative distance.

Given the above four conditions, the searching process can be made more efficient by implementing the following steps.

1. The comparison of s_i with s_j is redundant, if $j = i \pm 1$, since these two residues are connected.
2. Only non-diagonal distances are computed thus avoiding the necessity for any floating point operations.
3. Use the following polarity property (given in Sec. 4.2 below) based on polarity and relative distance of a residue. This property eliminates the redundant comparisons because the various equidistance groups are further subdivided with respect to their polarities as explained below.

4.2. Polarity property

To make the search efficient, the polarity (sign) of the coordinates of residues is exploited for matching purposes. The Figure 2 below shows the polarity consideration for the residues with respect to s'_1 .The symbols (+, -,•) will be used to indicate polarity, where + and – denote the relative signs of (x, y) with respect to s'_1 and • defines no polarity i.e. it will be matched as *don't care* provided the polarity of the other coordinate of a residue matches exactly.

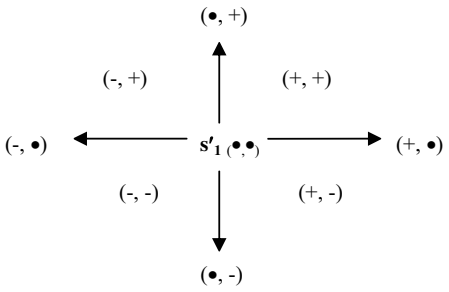


Figure 2. Diagram showing polarity consideration with respect to s'_1.

Thus both (+, -) and (+, •) as well as (+,+) and (+,•) are matchable while (+,•) and (•,+) or, (+,•) and (•, -) are examples of non matchable pairs. Similar pairs, for example (+, +) and (+, +) or, (-, +) and (-, +) are always matchable. In other words, it can also be seen from Figure 2, residues with similar polarity will match with each other as well as with those residues which are located at its two adjacent positions.

4.3. Scheme for polarity encoding

For computational ease, an encoding scheme for identifying polarity is also implemented. As shown in Figure 3, the polarities are encoded as three binary numbers. Two residues are considered matched when the encoding binary number of a residue finds either a same matching number or any of its two adjacent neighbours for another residue. For example, 010 is matchable with 010 and also with its two adjacent neighbours 011 and 001. By adding 001, the anti-clockwise immediate neighbour is found and by subtracting 001, the immediate clockwise neighbour is found. The operation is basically MOD 2 addition and subtraction.

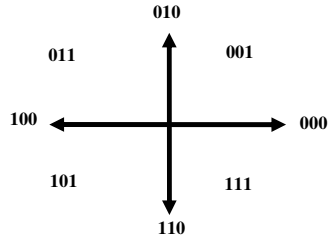


Figure 3. A three bit binary encoding for polarity.

4.4. Efficient search (ES) approach

In the ES approach, the relative distance and polarities of all hydrophobic residues with respect to s'_1 are calculated during the first scan. Let d_i be the distance of the i^{th} residue from s'_1. Then for any two residues s'_i and s'_j; there will be an H-H match if $|d_i - d_j|=1$ and also if the polarity of s'_i and s'_j is matched. The steps are summarized in the following Algorithm 2.

The following Table 2 is basically derived from Table 1, by considering the polarity of the (x, y) coordinates of $s'_1 \in S'$. For those values where either x=0 or y=0, the polarity is counted as '•' instead of '+'. Hence, in Table 2, it is observed that 6 (•, +) and 3 (+, +) have a match, while 6 and 7 do not since they are connected. 6 (•, +) and 15 (-, •) are also not matched because of their polarity mismatch. Similarly 20 (-,•) and 3 (+, +) are not matchable, while 20 (-,•) and 7 (-, +) are matchable. A similar procedure is followed for all subsequent levels. Note, for those residues where $d_i=1$, there is a direct match with the

starting residue 1 (i.e. s'_1) without the requirement for polarity matching. Hence (1, 6) and (1, 20) will have matches.

Algorithm 2. Efficient search approach

Precondition:
Fitness function, F=0; S′ (= s'_1, s'_2, s'_3 …, s'_n);
Coordinates of the hydrophobic residues of S′

Post condition: Fitness value F

1. FOR i (2 : n) DO
2. IF the distance between s'_1 and s'_i =1 THEN,
3. Form equidistant subsets based on residues which are equidistant from s'_1 and which also have polarity match
4. ENDIF
5. ENDFOR
6. WHILE match exists
7. Count the number of distance and polarity matches
8. Decrement F for each match
9. ENDWHILE

Table 2. Relative distance with polarity

Row 2, 3: (Relative) Polarity of the residues.

1	3	6	7	9	12	14	15	18	20
•	+	•	-	-	-	-	-	-	-
•	+	+	+	+	+	+	•	-	•
	2	1	2	4	5	3	2	3	1

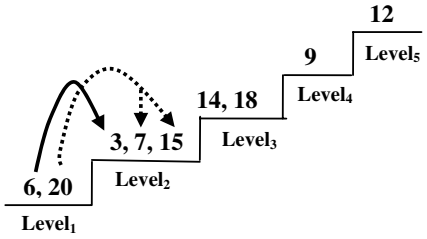


Figure 4. Residue match illustrated with the concept of levels.

The matching is illustrated in Figure 4 with the help of concept of levels in the 'Levels diagram'. In this diagram, the relative equidistant residues are represented at same 'Levels'. The i^{th} level is defined as $Level_i = d_i$. Thus, if the distance d of a residue (e.g. 6 or 20) is 1, that residue is considered at $Level_1$. The residues at one particular level should only be compared with the level immediately above it and will have a match if their polarities match with any of the residues in the level above. Hence, comparing 6 at $Level_1$ with 3, 7 and 15 at $Level_2$, we find that 6 will only match 3 but 6 will not match 7 and 15 due to the polarity mismatch. In Figure 4 below, the match between 6 and 3 is indicated by a solid arrow while the match of 20 with 7 and 15 is indicated by dotted arrow.

4.5. Missing levels

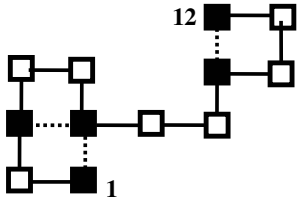


Figure 5. HP model of the sequence [101001001001]

Table 3. Coordinates and distance of hydrophobic residues.

Row 1: Hydrophobic residue index obtained from Figure 5.

	1	3	6	9	12
X	0	-1	0	2	2
Y	0	1	1	2	3
		2	1	4	5

Row 4: Relative lattice distance respect to residue at (0, 0).

The ES approach is valid even under a special case where a protein sequence may sometime result in an orientation that has missing levels. For example, we can have a condition where a level (say Level3 is missing). As an illustrative example, consider a binary string sequence, S= [101001001001] for a protein sequence. Figure 5 gives the HP model of protein folding orientation while Table 3 gives the coordinates and relative distances of the hydrophobic residues. It is observes from Figure 5 and Table 3, the presence of only three TN. This is because residue 1 has one neighbour (namely 6), residue 3 has one neighbour (namely 6) and residue 9 has one neighbour (namely 12).

The corresponding levels diagram is shown in Figure 6. Unlike just one 'stair' obtained in Figure 4 to illustrate the levels of residues, the levels diagram shown in Figure 6 clearly shows the existence of two separate stairs. This is due to the missing Level3 Residue 1 now matches with 3 directly. Also, 3 and 6 also match due to their adjacent polarity. 9 and 12 have a match since they have same polarity. Residue 6 will

not match with 9 because they are residues from different stairs. Hence the proposed ES approach is still valid in these special circumstances. The algorithm can ensure that there is no need to compare the residue of one stair with that of the other.

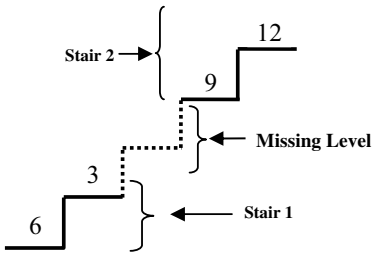


Figure 6. Missing levels resulting in two separate isolated stairs.

5. Complexity analysis

Let us consider a sequence with the total number of hydrophobic residues being *n*. The FS approach is given by Algorithm 1. It can be easily shown from Algorithm 1 that the time complexity of this approach is $\boldsymbol{O}(n^2)$.

Again, the proposed ES approach is summarized by Algorithm 2. We can observe that in steps 1 to 5, the relative lattice distances and polarities are computed by comparing (n-1) residues with respect to s'_1. In steps 6 to 9, all matching are counted. Let us consider a worst case scenario when the computation time is maximum. This happens when there will be *n* matches. In this case from equation (1), $M = 2 * \min(E[S], O[S])$. If it is assumed that, $E[S] = O[S] = n/2$, then, $M = 2 * (n/2) = n$.

Also, assume that, there is no $d_i = 1$, i.e. no match is found at the initial scan. Hence, there will be a maximum of only $(n-1)+n = (2n-1)$ comparisons, so the resulting time complexity is $\boldsymbol{O}(n)$. This is a significant improvement when compared with the time complexity of $\boldsymbol{O}(n^2)$ for FS.

6. Conclusions

This paper presents a new efficient search (ES) method for fitness computation. In comparison to the full search approach, ES eliminates redundant comparisons and exploits the intrinsic relationship between *odd-1* and *even-1* grouping with respect to the HP model. The time complexity of ES is $\boldsymbol{O}(n)$ compared with $\boldsymbol{O}(n^2)$ for FS. This improvement in time complexity is significant in terms of being able to speedily compute protein folding steps. For example, when using genetic algorithms for the protein folding problem, the fitness function is often used to test the fitness of a particular amino acid chain sequence.

7. References

[1] Allen, et al., "Blue Gene: A vision for protein science using a petaflop supercomputer", *IBM System Journal*, 2001, VOL 40, No 2.

[2] Berger, B. and Leighton, T., "Protein Folding in the Hydrophobic-Hydrophilic (HP) model is NP-Complete", *ACM, Proceedings of the second annual international conference on Computational molecular biology*, 1998.

[3] Dill, K.A., "Theory for the Folding and Stability of Globular Proteins", *Biochemistry*, 1985, 24: 1501.

[4] Gary, B.F. and David, W.C. (Editors), *Evolutionary Computation in Bioinformatics*, Elsevier Science (USA), 2003.

[5] Hart, E.W. and Istrail, S., "Fast Protein Folding in the Hydrophobic-hydrophilic Model Within Three-eights of Optimal", *ACM*, 1995.

[6] König, R. and Dandekar, T., "Refined Genetic Algorithm Simulation to Model Proteins", *Journal of Molecular Modeling*, 1999.

[7] Lamont, G.B. and Merkie, L.D., "Toward effective polypeptide chain prediction with parallel fast messy genetic algorithms", *Evolutionary Computation in Bioinformatics,* edited by Gary Fogel and David Corne, Chap7, pp. 137- 161.

[8] Mauri, G., Piccolboni, A. and Pavesi, G., "Approximation Algorithms for Protein Folding Prediction", *ACM*, 1999.

[9] Newman, A., "A new algorithm for protein folding in the HP model", *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete Algorithms*, 2002.

[10] Rune, B.L., Christian, N.S. and Pedersen, "Protein Folding in the 2D HP model", *Bioinformatics 2002, http://www.daimi.au.dk/~cstorm/bioinf_f02/misc/jobim00_hp.pdf* , 2002.

[11] Takahashi, O., Kita, H. and Kobayashi, S.,"Protein Folding by A Hierarchical Genetic Algorithm", *the 4th Int. Symp. On Artificial Life and Robotics (AROB)*, 1999.

[12] Yap, A. and Cosic, I., "Application of Genetic Algorithm for Predicting Tertiary Structure of Peptide Chains", *IEEE*, 1999.