FAST COMPUTATION OF THE FITNESS FUNCTION FOR PROTEIN FOLDING PREDICTION IN A 2D HYDROPHILIC-HYDROPHOBIC MODEL

MD. TAMJIDUL HOQUE, MADHU CHETTY AND LAURENCE S DOOLEY

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

Abstract: Protein Folding Prediction (PFP) is essentially an energy minimization problem formalised by the definition of a fitness function. Several PFP models have been proposed including the Hydrophobic-Hydrophilic (HP) model, which is widely used as a test-bed for evaluating new algorithms. The calculation of the fitness is the major computational task in determining the native conformation of a protein in the HP model and this paper presents a new efficient search algorithm (ESA) for deriving the fitness value requiring only $O(n)$ complexity in contrast to the full search approach, which takes $O(n^2)$. The improved efficiency of ESA is achieved by exploiting some intrinsic properties of the HP model, with a resulting reduction of more than 50% in the overall time complexity when compared with the previously reported *Caching Approach,* with the added benefit that the additional space complexity is linear instead of quadratic.

Keywords: HP Model, Fitness Function, Improved Computation, Relative Distance and Polarity.

1 INTRODUCTION

Proteins are the fundamental components of all living cells, with protein misfolding being recognised as a cause behind such diseases as Alzheimer's disease, Mad cow problem, Parkinson's disease, new variant CJD and type II diabetes [Goldberg, 2004]. To make any protein, ribosomes form a linear sequence of different amino acids, each taken from a codebook of 20 unique amino acids. This one-dimensional chain is then converted into a three dimensional shape called its *native conformation*, which provides an insight into that particular protein's functionality, and this has been one of the primary foundations for research into *Protein Folding Prediction* (PFP). The National Grand Challenge in bio-chemistry in the United States [Lamont and Merkie, 2003] previously identified both the importance and the very computationally intensive nature of this problem, with research not only being directed towards determining the in-vivo structures of naturally occurring proteins, but also promoting protein design. For any requisite fold, the corresponding amino acid sequence has to be predicted, a challenge commonly referred to as the *inverse protein folding problem* [Gupta et al, 2004], which is now the focus of research in the drug design area.

The native conformation of a protein is determined by the influence of several regular forces [Rune et al, 1999] applied to the amino acid sequence. Amino acids are categorized as being either positively or negatively charged, and then based on side chain size; they are further sub-divided as tiny, small and large, even aliphatic or aromatic and so on. One feature that significantly impacts upon protein folding is *hydrophobicity*, which governs how amino acid residues are to be classified. The two categories are; i) *Hydrophobic* (H) or *non-polar* residues which are repelled by water [Allen et al, 2001], and tend to be inside the protein core; and ii) *Hydrophilic* or *polar* (P) residues which are attracted to water and tend to remain outside the protein core. These two components are the kernel blocks of the Hydrophobic-Hydrophilic (HP) model [Dill 1985] which is widely used in PFP applications. Moreover, the usage of HP model for *inverse protein folding problem* has been established recently [Gupta et al, 2004].

The native conformation for any amino acids chain is the conformation with the lowest energy and it is achieved when the numbers of hydrophobic-hydrophobic (H-H) pairs, referred to as topological neighbour (TN), is a maximum [Fogel and Corne, 2003]. By definition, a TN is an adjacent H pair that is a unit lattice distance apart, with the proviso, that those that are sequential with respect to the formed chain are excluded.

While the HP model is widely used as an empirical vehicle, even a simplified PFP model incurs a high computational cost. For instance, for an amino acid

chain of 150 means, the number of possible conformations becomes enormous since the total number will be n^{150} assuming n degrees of freedom. Perhaps not surprisingly, finding the conformation with the minimum energy in a 2D HP model has been proven to be a NP-complete problem [Crescenzi et al, 1998], so a non-deterministic search strategy needs to be employed. Previous techniques that have been used include Monte Carlo (MC), Genetic Algorithm (GA) [König and Dandekar, 1999; Takahashi et al, 1999; Unger and Moult, 1993a and 1993b; Yap and Cosic 1999], Evolutionary MC (EMC) [Bastolla et al, 1998; Liang and Wong, 2001], Simulated Annealing (SA), and Tabu Search with GA (GTB) [Jiang et al, 2003]. All these approaches are characterised to some extent by essentially being a random search with clues, yet they still incur a high computational overhead, with more iterations leading to a greater likelihood of achieving an optimum or near optimum solution for a given amount of time. Many approximation algorithms [Hart and Istrail, 1995; Mauri et al, 1999; Newmann, 2002] have also been developed to ensure faster protein folding computation, though exact prediction still remains an elusive goal, which provided the main motivation for the strategies presented in this paper to improve resource efficiency i.e., computational throughput.

The major objective for PFP in the 2D HP model [Santos and Santos, 2001 and 2004] is to manage the very large number of search operations as efficiently as possible, without compromising prediction accuracy. While the fitness function *per se* in HP model is simple, the aim is to limit the total number of samples tested to provide the best fitness value within a prescribed time interval. Optimization of the fitness computation has often been neglected, which is odd given that repeated fitness computations form a major component of the search process in non-deterministic approaches. There is a distinct absence of literature upon how to efficiently compute the fitness function, with the notable exception of [Hoque et al, 2004] which is analysed and compared later in the paper. An alterative strategy in [Santos and Santos, 2001 and 2004] proposed the use of a cache to reduce the computational load, though this incurred a higher time and memory overhead in comparison to the technique proposed by Hoque et al. in [Hoque et al, 2004]. This paper presents a new *Efficient Search Algorithm* (ESA) in respect to both time and space complexity. The time complexity of the Full Search Algorithm (FSA) is quadratic, whereas ESA has linear complexity and also requires less than 50% of the number of operations for fitness computation, when compared with the *cache-based* approach reported in [Santos and Santos, 2001]. Also, additional space complexity in ESA is linear, whereas it is quadratic in [Santos and Santos, 2001].

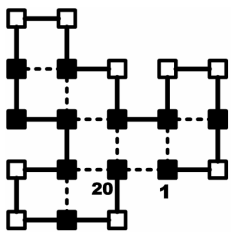
The remainder of the paper is organized as follows. In Section 2, the HP model and formulation of amino acid chain string is described, while Section 3 describes the FSA approach to computing the fitness value and Section 4 reviews the Caching Approach. Section 5 details the theoretical basis for ESA with a series of lemma proving the reasons for the improved computational performance, while Section 6 presents a computational complexity analysis. Section 7 describes the experimental results. Finally, some conclusions are given in Section 8.

2 HP MODEL

For any given amino acid sequence, a number of valid conformations are possible. A valid conformation has self-avoiding walk in the lattice model. The desired conformation is the one that has maximum number of TNs, so when searching, the higher the number of TNs, the lower the fitness function, and the closer that conformation is to the desired folding.

2.1. HP Model and Fitness Function

In a 2D HP model, the conformation is represented by placing the amino acid chain on a square lattice model. A conformation with a self-avoiding walk is a valid conformation; otherwise it is an invalid conformation. Figure 1 shows an HP model example for a fitness value of -9, where the hydrophobic and hydrophilic residues are represented by black and white squares respectively. A solid line connecting two squares indicates concatenated amino acids, while the dotted line indicates a TN pair.



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

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

The following approach, given in [Fogel and Corne 2003], has been used in the proposed work to measure 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 chain string S is traversed to determine the number of TN pairs in the HP model. From Figure 1, it is clear there are 9 such pairs so the fitness function value is -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). Also for presentation ease, the hydrophobic and hydrophilic residues are represented as binary '1' and binary '0' respectively.

2.2. Binary String Formulation

The chain of amino acid sequence in Figure 1 can be represented as $S = [10100110100101100101]$. A binary '1' at an odd and even index is respectively referred to as *odd-1* and *even-1* [Newman, 2002].

Table 1: Coordinates and relative lattice distance.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
s_i'	**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
d_i	0	2	1	2	4	5	3	2	3	1

For any string, there is a fixed range of values of the fitness function, F given as $0, -1, -2, \ldots, -M$. The maximum value M being [Newman, 2002]:

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

where $O[S]$ and $E[S]$ are the number of *odd-1* and *even-1* in the string respectively and it is assumed that neither of the end points in the string are hydrophobic residues. If any end point is hydrophobic then one additional TN is possible. Therefore, the upper bound for (1) can be expressed [Rune et al, 1999] as,

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

It is clear that on a square lattice, an *even-1* will always be adjacent to *odd-1*. Hence, each element in the string S can have a maximum of two TNs whereas the residue at the end position can have maximum of three TNs.

The string $S \in \{0,1\}^m$ can be represented as binary string, $S = [s_1, s_2, s_3, \cdots, s_m]$. Let us consider S' to be the string having n hydrophobic residues only from S and S' is an ordered number set holding index i of s_i where s_i has value '1' and $n \le m$. Let, $S' = [s_1', s_2', s_3', \cdots, s_n']$. The relative lattice distance, d_i is measured from s_1' to any s_i', $d_i = |x_i| + |y_i|$, where $(1 \le i \le n)$. The values of d_i for various residues are shown in the last row of Table 1. For example, d_6 in Table 1 is 5 which corresponds to the 12th hydrophobic residue.

3 FULL SEARCH ALGORITHM (FSA)

This algorithm computes the fitness function F, by comparing s_1' is firstly 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 FSA are given in Algorithm 1 below.

Algorithm 1: Full search algorithm

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 number of hydrophobic residues is *n*, the computation of F takes $O(n^2)$ time complexity.

4 CACHING APPROACH

Another approach reported in [Santos and Santos, 2001 and 2004], uses cache for reducing the computational load. For convenience it will be referred to as the *Caching Approach* [Santos and Santos, 2001] and involves the full chain sequence

being re-mapped into a $m \times m$ memory *Matrix* or, *Grid* G, where m is the total number of residues in the chain. As shown in Table 1, a 2D array $R(x, y)$ contains the coordinates of i^{th} residue, i.e.

$$R[i][1] = x_i \text{ and } R[i][2] = y_i \quad (3)$$

With the chain length of m residues, it is re-mapped into G as

$$G[1][1] = \min(x_i) \text{ and } G[1][2] = \min(y_i) \quad (4)$$
$$\text{where, } 1 \le i \le m .$$

Now consider

$$a \leftarrow (R[i][1] - x_{\min}) \text{ and } b \leftarrow (R[i][2] - y_{\min}) \quad (5)$$

$G[a][b]$ is assigned either 'H' or 'P' depending on whether the i^{th} residue is hydrophobic or hydrophilic, respectively. For a 'H' in the cell of G, four neighbours: $(a+1, b)$, $(a, b+1)$, $(a-1, b)$ and $(a, b-1)$ are examined as a possible TN match. Therefore, for the total of n hydrophobic residues, there will need to be $4n$ comparisons. Moreover, this approach keeps track of hydrophilic residue, which means further $(m-n)$ comparisons. So, total comparisons are $(4n + (m-n))$ or $(3n+m)$. For the $4n$ lookups, there must be a guard memory of 1 cell width around the $m \times m$ grid, which requires $(4m+4)$ guard memories. Finally, the total additional memory requirement for the *Caching Approach* is

$$(m^2 + 4m + 4) \quad (6)$$

5 REDUCING THE COMPLEXITY

By considering the even and odd index positions of a '1' in the string S, groupings can be formed of either *even-1* or *odd-1* according to their index number. The orientation of the members of these groups elicits some very useful properties which can be exploited to reduce the time complexity.

5.1. Towards an Efficient Search Algorithm

The following lemmas are presented as the theoretical basis for constructing the new efficient search algorithm (ESA).

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

Proof: In a square lattice, an *odd-1* can only be adjacent to *even-1* and vice versa. Adjacent *odd-1*s 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 to any *even-1* will always differ by an 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.

1. 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.
2. Using Lemma 2, if a particular point is *odd-1* then all *even-1*s will be odd distanced from that point and all *odd-1*s will be even distanced.
3. Using Lemma 3, for any hydrophobic residue, some *odd-1s* and some *even-1s* are alternatively separated on the basis of relative distance.
4. 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.

By exploiting these four propositions, the search process can now be implemented more efficiently as follows:

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 in Section 5.2 based on sign and relative distance of a residue, redundant comparisons can be eliminated because the various equi-distance groups are further subdivided with respect to their signs or polarities.

5.2. 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 $(+, -, \bullet)$ are 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.

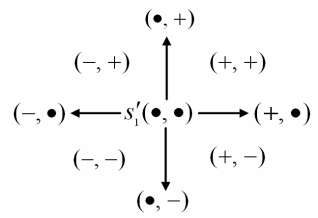


Figure 2: Polarity consideration for residues with respect to s′1.

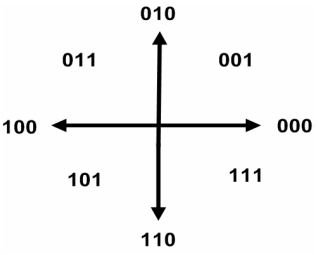


Figure 3: A three bit binary encoding for polarity.

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

5.3. Scheme for Polarity Encoding

For computational ease, an encoding scheme for identifying polarity is also implemented. As Figure 3 illustrates, the polarities are encoded as three bits. 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* not only matches itself, but 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, so the operation is a *MOD 2* addition and subtraction.

5.4. Efficient Search Algorithm (ESA)

In the ESA 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.

Algorithm 2: Efficient search algorithm.

Precondition:
Fitness function, F=0; $S'(= s_1', s_2', s_3' \ldots, s_n')$;
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. FOR i (2 : n) DO
7. Count the number matches found
8. Decrement F for each match
9. ENDFOR

Table 2 is basically derived from Table 1, by considering the polarity of the (x, y) coordinates. 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 $6(\bullet, +)$ and $3(+, +)$ have a match, while 6 and 7 do not since they are connected. $6(\bullet, +)$ and $15(-, \bullet)$ are also not matched because of their polarity mismatches, as are $20(-, \bullet)$ and $3(+, +)$, while $20(-, \bullet)$ 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 i.e. s_1' without the requirement for polarity matching. Hence $(1, 6)$ and $(1, 20)$ will have matches.

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

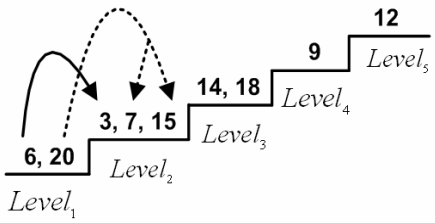


Figure 4: Residue match illustrated with the concept of levels.

This matching is illustrated in Figure 4 where the concepts of levels and a *Level Diagram* are introduced. In this diagram, the relative equidistant residues are represented at the same *Level*. 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 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 only matches 3 but 6 will not match 7 and 15 due to the polarity mismatch. In Figure 4, 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.

5.5. Missing Levels

ESA is robust enough so that it is valid even under the special circumstances where a protein sequence sometimes results in an orientation that has missing levels. Consider for example the condition where $Level_3$ is missing in a binary string sequence, $S = [1010010010\ 01]$. Figure 5 shows the corresponding conformation in the HP model, and Table 3 gives the coordinates and relative distances of the hydrophobic residues, from which it can observed that there are only three TN. This is because residues 1, 3 and 9 all have just a single neighbour, namely 6, 6 and 12 respectively.

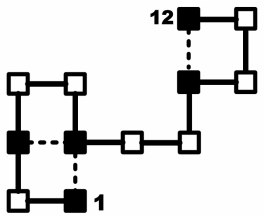


Figure 5. HP model of the sequence [101001001001]

Table 3: Coordinates and distance of hydrophobic residues.

i	1	2	3	4	5
s_i'	1	3	6	9	12
x	0	-1	0	2	2
y	0	1	1	2	3
d_i	0	2	1	4	5

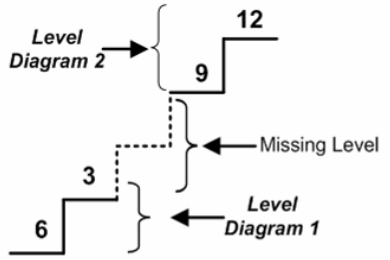


Figure 6: Missing levels resulting in two separate isolated stairs.

The corresponding *Level Diagram* is shown in Figure 6, which unlike that in Figure 4 actually reveals the existence of two separate *Level Diagrams* due to $Level_3$ being missing. Now, residue 1 matches with 6 residues directly. Also, 3 and 6 match due to their adjacent polarity. 9 and 12 have a match since they have same polarity. Residue 6 will not match however with 9 because they are residues in different *Level Diagrams*. This confirms that the new proposed ESA is still valid in these special

circumstances, with there being no need to compare the residue in one diagram with that of another.

5.6. Data Structure and Implementation of ESA

For implementing ESA, a $n \times 8$ grid of memory is allocated. The 8 columns represent the indices from 0 to 7 in order, which correspond to the encoded polarity values described in section 5.3. Using the same tabular format as in Table 2, the $(n-1)$ residues, excluding the starting residue, are re-mapped into the grid to facilitate fitness calculation. Further, it is assumed that each row is circular, so column 0 and 7 for example are adjacent. To traverse any row, two simple functions, namely *next* and *previous* are defined in the context of column numbering, so the *previous* column 7 is 6, while the *next* of column 7 is 0, or the *next* column of 0 is 1, so on. That is, for i^{th} column the next $(i+1)^{th}$ column is calculated by the formula: $\{(i+1) \bmod 8\}$, while to find the *previous* $(i-1)^{th}$ column, least significant 3 bits from the result of $(i-1)$ are taken. To conceptualise this structure, the memory can be considered as being a cylindric configuration whose thickness equals 1 cell, perimeter had 8 cells and height has n cells. For the sake of clarity, the *Fitness Calculator Grid* (FCG) is represented as a $n \times 8$ grid in Figure 7.

Algorithm 3: Extended ESA.

BEGIN
1. *Initial values of all the cells of row 1 are set 1 and all others cells are set 0.*
2. *FOR i (2: n) DO*
   $x \leftarrow d_i$ */* Refer Table 2 */*
   $y \leftarrow$ *encoded value of polarity*
   $FCG(x+1, y) \leftarrow FCG(x+1, y)+1$
   $FCG(x+1, next(y)) \leftarrow FCG(x+1, next(y))+1$
   $FCG(x+1, previous(y)) \leftarrow FCG(x+1, previous(y))+1$
   ENDFOR
3. $MatchCount \leftarrow 0$
   FOR i (2: n) DO
   $x \leftarrow d_i$
   $y \leftarrow$ *encoded value of polarity*
   $MatchCount \leftarrow MatchCount + FCG(x, y)$
   ENDFOR
4. $MatchCount \leftarrow MatchCount$ *– (Total number of concatenated H-H pairs)*
   $F \leftarrow (-1) * MatchCount$
END.

To incorporate the above property of encoding with distance for the purpose of fitness computation, the ESA presented in *Algorithm 2* has been modified to become the *Extended* ESA given in Algorithm 3.

The 'Total number of concatenated H-H pairs' used in *Step 4* is calculated once and remain constant for all the valid conformations of a particular chain. For example, in Figure 1 since $(6, 7)$ and $(14, 15)$ are the two concatenated H-H pairs, the total will be 2. To minimize the amount of checking, the proposed approach does not differentiate between H-H interactions with TNs, with the number of actual TNs being found by deducting the number of concatenated H-H pairs count from the total match count.

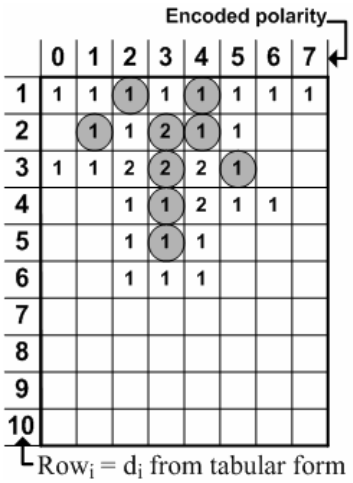


Figure 7: Fitness Calculator Grid (FCG), demonstrating fitness computation.

The data structure and implementation can be illustrated by the example in Figure 7. Table 2 is re-mapped for fitness computation in a FCG, shown in Figure 7, where cell values of 0 are shown as blank for the sake of simplicity. With first pass of the tabular form of $(n-1)$ elements, the inputs of the cells of FCG are given and updated. With second pass of the tabular form of $(n-1)$ elements, matches are computed using FCG cell values. The computed matches are shown marked by dark circles, and their summation in the example is 11. Since the 2 concatenated pairs of H-H is included, the exact TN count will be $(11-2)=9$. So, the fitness $F=-9$.

Generalising this example, it is clear that the number of operations will be $(n+1)+(n+1)=(2n-2)$ and the space requirement is $8n$.

6 COMPLEXITY ANALYSIS

In order to analyse the improvement in the computational efficiency of ESA, consider a sequence of m residues, with n hydrophobic residues where, $m \geq n$ for obvious reason. The FSA is given by Algorithm 1. The time complexity of FSA (*Algorithm 1*) is $O(n^2)$, while for *Caching Approach*, the number of operations are $(3n+m)$ and a lower bound of $4n$ operations where, $m = n$.

The computational performance of the new ESA (*Algorithm 2*) requires in *Steps 1* to *5*, the relative lattice distances and polarities to be computed by comparing $(n-1)$ residues with respect to s_1'. In *Steps 6* to *9*, all the matches are counted and from Section 5.6, it is clear that the $(n-1)$ elements are accessed twice in *Step 2* and *3*. The order of time complexity is $O(2n\text{-}2) = O(n)$. With respect to quadratic time complexity of FSA, ESA has linear time complexity. Compared to *Caching Approach*, the total saving in the number of operations in ESA that determines the lower bound (LB) is,

$$\eta_{T_{LB}} = \left(1 - \frac{2n-2}{4n}\right) \tag{7}$$

Simplifying (7), we can write it as,

$$\eta_{T_{LB}} = \left(\frac{1}{2} + \frac{1}{2n}\right) \tag{8}$$

That is, the reduction in number of operations is more than 50%. The actual operations in the *Caching Approach* can be stated as,

$$(3n+m), \text{ where } (3n+m) \geq 4n \tag{9}$$

Therefore, generally speaking, the reduction in operations will be in reality, more than that stated in (8). The actual reduction will be,

$$\eta_T = \left(1 - \frac{2n-2}{3n+m}\right) \tag{11}$$

The average case distributation of n over m can be shown to be,

$$n = \frac{m}{2} \tag{12}$$

Using (12) in (11), the reduction can written as,

$$\eta_T = \left(1 - \frac{2n-2}{5n}\right) \tag{13}$$

Simplifying (13), we obtain,

$$\eta_T = \left(\frac{3}{5} + \frac{2}{5n}\right) \tag{14}$$

From Eqn. (14), it can be clearly seen that the resultant saving is significantly more than 50%.

For ESA, the space or memory requirement is $8n$. Therefore, the space complexity is $O(n)$ in contrast to the *Caching Approach* in which complexity is quadratic as shown by (6). Using (12) in (6), the average space requirement for *Caching Approach* can be written as,

$$(4n^2 + 8n + 4) \tag{15}$$

Clearly, the above expression is quadratic in nature.

7 EXPERIMENTAL RESULTS

In this Section, the time complexity of FSA, *Caching Approach* and ESA are compared. As the comparison is essentially based on number of operations required to compute fitness function, the pre-processing time for each of the methods has not been included for calculating the time requirement. Those operations (for example, for the initialization of a 2D array with all 0s) that are hardware or complier specific are not included as well. For the ESA, the operation can be summarised with only two operators namely *write* and *read* operations in FCG. In Step 2 of Algorithm 3, addition in three adjacent cells of (x, y) can be summarised as the *write* in cell (x, y). The *read* operation is used to read and add match count in Step 3. It must be noted that the three independent additions constituting *write* operation can be implemented in parallel, which will speedup ESA further. Table 4 shows the time comparison, where Algorithm 3 is used for ESA. The column with the heading as CA indicates the relative time requirement of *Caching Approach*. The value of n is taken to be the average of 1000 times random occurrence of hydrophobic residues for any particular number of the total m residues. Each method is invoked 1000 times to avoid the zero or near zero measures and the time is measured in tick-count.

Additional space requirement for *Caching Approach* and ESA are dependent on m and n respectively. The symbols m and n respectively represent the total number of residues and total number of hydrophobic residues in a sequence. From Table 4, it is clear that the average distributation of n is approximately 50%

of *m*, which supports the assumption given by (12). Therefore, for ESA the space requirement is $8n$ whereas for the *Caching Approach* the space requirement on the average is $(4n^2+8n+4)$, as expressed in (15).

Table 4: Time comparison of FSA, CA and ESA. Time is measured in 1000 tick-count, where 999 tick-counts are equal to 1 second in VB 6.0.

m	*n* (avg)	FSA	CA	ESA
20	9.98	30.65	49.26	34.03
60	30.09	329.73	148.17	109.65
100	49.69	941.47	249.64	186.15
140	69.56	1845.60	343.70	257.69
180	90.13	3127.35	443.42	335.34
220	110.00	4631.26	538.05	407.06
260	129.75	6449.18	633.96	481.17
300	150.04	8714.12	739.59	559.98
340	169.74	11318.65	838.49	634.24

8 CONCLUSIONS

This paper presents a new efficient search algorithm (ESA) for fitness computation. In comparison to the full search algorithm, ESA eliminates redundant comparisons and exploits the intrinsic relationship between various indexed groupings with respect to the HP model. The order of time complexity for ESA is *O(n)* compared to $O(n^2)$ for FSA. Compared to the *Caching Approach*, the time complexity is less. Further, the space complexity of ESA is linear whereas for *Caching Approach* it is quadratic. This improvement in time complexity is significant as it enables efficient computations of the protein folding prediction algorithm that uses, for example a genetic algorithm, since the fitness function is repeatedly computed to test the fitness of a particular amino acid chain sequence.

REFERENCES

[1] Allen, et al. 2001, "Blue Gene: A vision for protein science using a petaflop supercomputer", IBM System Journal, Vol 40, No. 2, pp.310-327.

[2] Bastolla, U., Frauenkron, H., Gerstner, E., Grassberger, P. and Nadler, W. 1998, "Testing a new Monte Carlo Algorithm for Protein Folding", National Center for Biotechnology Information, Vol. 32, No. 1, pp.52-66.

[3] Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A. and Yannakakis, M 1998, "On the complexity of protein folding (extended abstract)", ACM, Proceedings of the second annual international conference on Computational molecular biology. Pp.597-603.

[4] Dill, K.A. 1985, "Theory for the Folding and Stability of Globular Proteins", *Biochemistry*, Vol. 24, No. 6, pp.1501-1509.

[5] Fogel, G.B. and Corne, D.W. (Editors) 2003, Evolutionary Computation in Bioinformatics, Elsevier Science (USA).

[6] Goldberg A. 2004, "Introduction to protein folding and disease", © Nature Publishing Group, http://www.nature.com/horizon/proteinfolding/sessionsummaries.html

[7] Gupta A., Maňuch J. and Stacho L. 2004, "Inverse protein folding in 2D HP model", IEEE CBS'04 Computational Systems Bioinformatics Conference. Pp.311-318.

[8] Hart, E.W. and Istrail, S. 1995, "Fast Protein Folding in the Hydrophobic-hydrophilic Model Within Three-eights of Optimal", Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, Las Vegas, Nevada, United States. Pp.157-168.

[9] Hoque M.T., Chetty, M. and Dooley L.S. 2004, "An Efficient Algorithm for Computing the Fitness Function of a Hydrophobic-Hydrophilic Model". 4th International Conference on Hybrid Intelligent Systems (HIS'04). Pp.285-290.

[10] Jiang, T., Cui, Q., Shi, G. and Ma, S. 2003, "Protein Folding Simulations of the Hydrophobic-Hydrophilic Model by Combining Tabu Search with Genetic Algorithms", Journal of Chemical Physics, Vol. 119, No. 8, pp.4592-4596.

[11] König, R. and Dandekar, T. 1999, "Refined Genetic Algorithm Simulation to Model Proteins", Journal of Molecular Modeling, Vol. 5, pp.317-324.

[12] Lamont, G.B. and Merkie, L.D. 2003, "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.

[13] Liang, F. and Wong, W.H. 2001, "Evolutionary Monte Carlo for protein folding simulations", Journal of Chemical Physics, Vol. 115, No. 7, pp.3374-3380.

[14] Mauri, G., Piccolboni, A. and Pavesi, G. 1999, "Approximation Algorithms for Protein Folding Prediction", Symposium on Discrete Algorithms (SODA), Baltimore, Maryland, USA. Pp.945-946.

[15] Newman, A. 2002, "A new algorithm for protein folding in the HP model", Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete Algorithms, San Francisco, California. Pp.876-884.

[16] Rune, B.L., Christian, N.S. Pedersen 1999, "Protein Folding in the 2D HP model", http://www.brics.dk/RS/99/16/BRICS-RS-99-16.pdf, Basic Research in Computer Science (BRICS). Pp.1-15.

[17] Santos, E.E. and Santos, E.J. 2001, "Effective and Efficient Caching in Genetic Algorithms", International Journal on Artificial Intelligence Tools, Vol. 10, No. 1-2, pp.273-301.

[18] Santos, E.E. and Santos, E.J. 2004, "Reducing the Computational Load of Energy Evaluations for Protein Folding", Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04). Pp.79-86.

[19] Takahashi, O., Kita, H. and Kobayashi, S. 1999,"Protein Folding by A Hierarchical Genetic Algorithm", the 4th Int. Symp. On Artificial Life and Robotics (AROB99). Pp.334-339.

[20] Unger, R and Moult, J. 1993a, "Genetic Algorithms for Protein Folding Simulations", Journal of Molecular Biology, Vol. 231, pp. 75-81.

[21] Unger, R. and Moult, J. 1993b, "On the Applicability of Genetic Algorithms to Protein Folding", Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences, Vol. 1, pp.715-725.

[22] Yap, A. and Cosic, I. 1999, "Application of Genetic Algorithm for Predicting Tertiary Structure of Peptide Chains", Proceedings of First Joint Biomedical Engineering Soc. (BMES)/ Engineering in Medicine and Biology (EMBS), Atlanta, GA, USA. Vol. 1, p.1214.

AUTHORS' BIOGRAPHIES

MD TAMJIDUL HOQUE (Student Member, IEEE) received his B. Sc. Engg. (Hons) and M. Sc. Engg. degrees in Computer Science and Engineering (CSE) from Bangladesh University of Engineering and Technology (BUET) in 1998 and 2002 respectively. He was a lecturer at the CSE department, Ahsanullah University of Science and Technology (AUST) from September 1998 to December 1999. He was in technical management as IT departmental head and DGM in Bashundhara Group from December 1999 to March 2004. Since April 2004, he is a Ph.D. student at GSIT, Monash University, Australia. His current research interest is in bioinformatics with focus on *ab initio* Protein Folding Prediction. His other research interests include Networking and Communication, Database System, Complier Design, Automata Theory, Distributed Systems and Parallel Computing, Algorithms, Computer Architecture, Petri Net Theory, Security and Operating System.

MADHU CHETTY

Madhu R Chetty (M'95—SM'03) received his B.E.(Elec), M.Tech. and Ph.D. degrees in Electrical Engineering from the Nagpur University (India) in 1976, 1980 and 1992 respectively.

Since 1980, he has been involved in teaching, research and sponsored industry consultancy while working at various universities in India, The Netherlands and Australia. He is with Monash University, Australia since 1995 with major research interests in bioinformatics, intelligent control, complex systems, optimisation, grid computing and haptic exploration in virtual environment. He has published numerous papers in refereed international conferences and journals. He has been recipient of various research grants from Monash University to support his research. He is currently Senior Lecturer and the Course Director of the undergraduate and honours program at the Gippsland School of Information Technology, Monash University.

Dr. Chetty is a Senior Member of IEEE and Fellow of Institution of Engineers (India). He is member of the Victorian Bioinformatics Consortium, Melbourne and the Bioengineering and Bioinformatics technical committee of IEEE Computational Intelligence Society.

LAURENCE S DOOLEY

Laurence S. Dooley (M'81—SM'93) received his B.Sc.(Hons), M.Sc. and Ph.D. degrees in Electrical and Electronic Engineering from the University of Wales, Swansea in 1981, 1983 and 1987 respectively.

Since 1999, he has been Professor of Multimedia Technology in the Gippsland School of Information Technology, Monash University, Australia, where his major research interests are in multimedia signal processing, shape-based video object coding, bioinformatics, wireless sensor networks and mobile communications. He has published well over 130 international scientific peer-reviewed journals, book chapters and conference papers, and successfully supervised thirteen PhD/Masters research degrees. He is also currently Executive Director of the Monash Regional Centre for Information and Communications Technology, which has established a Technology Transfer Gateway Network (TTGN) to stimulate the innovation cycle of regional small enterprises, and provide a bridge to commercialisation of new emerging technologies for its industry partners. The TTGN project is partly funded by a Commonwealth AusIndustry *Innovation Access* grant.

Professor Dooley is a Senior Member of the IEEE, a Chartered Engineer (C.Eng), and a full member of the British Computer Society (MBCS) and CITP.