Partially Computed Fitness Function Based Genetic Algorithm for 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

Fitness computation after each crossover or mutation operation in Genetic Algorithm (GA) requires computational time that increases with the increasing length of the chromosome. In this paper, an efficient GA is proposed for protein folding prediction based on the Hydrophobic-Hydrophilic (HP) model. The partial fitness of the parent computed from one end of sequence till crossover or mutation point is utilized for the computation of the fitness of the child. The calculated value of the partial fitness is stored with the corresponding chromosome. Although the approach requires additional memory for each hydrophobic residue of each chromosome, the computation time is reduced significantly which is more important than the memory overhead.

Keywords: Partial fitness, genetic algorithm, protein folding, hydrophobic-hydrophilic residue.

1. Introduction

The prediction of protein folding pathway from a sequence of amino acid is still an unsolved problem [8]. There are several forces affecting the protein folding [5]. Some forces are very strong and others are weak. *Hydrophobicity* is one of the strongest forces. Based on this force groupings of hydrophobic (H) or non-polar and hydrophilic or polar (P) is done [1]. H residues tend to form the protein core as they are repelled by water. But P residues are attracted to water and tend to be outside of the protein core. Based on this property, the HP model was introduced [3]. In a 2D HP model, a possible conformation is represented by placing the amino acid chain on a square lattice model having a self-avoiding walk. Sequence (A) of Figure 1 shows an HP model having a valid (self-avoiding walk) conformation.

Protein folding prediction in the 2D HP model being a NP-complete problem [2], a randomised search scheme like Genetic Algorithm (GA) is used for the HP model [4][7][9][10]. New solutions are achieved in GA through the operations of selection, crossover and/or mutation. The genetic algorithm requires large numbers of samples from the solution space and each new solution has to be evaluated by computing its fitness. This approach needs significant computational time that increases with the increasing length of the chromosome. Thus, any improvement in the fitness computation will increase significantly the total number of solutions that can be evaluated within a given time. In this paper, we show how efficiency increases by storing the partial fitness cumulative sum in an ordered traversal of the conformation. After crossover or mutation for the new conformation, fitness computation is made more efficient by utilizing partial fitness of its ancestor.

The remainder of the paper has been organized as follows. In section 2, efficient fitness computation has been discussed. Section 3 gives the mathematical analysis. Crossover operation and partial fitness computation are analyzed in section 4, while section 5 deals mutation. Simulation results are given in section 6. Finally, section 7 concludes the work.

2. Efficient fitness computation

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, is that adjacent H pair that has unit lattice distance but is not sequential or is not connected. For fitness computation TN counting is started from

hydrophobic residue number one and we attempt to find its four TN by comparing with other residues. Next, we go for the second hydrophobic residue and search for any of its TN traversing this way till the last hydrophobic residue is reached. The fitness is then equal to the total number of TN encountered multiplied by (-1). A crossover example [9] is shown in Figure 1 along with the fitness function computation. A similar [9] example for mutation is shown in Figure 3. Whether there is a crossover or a mutation, the fitness has to be computed each time. As the fitness computation is a very frequent operation, optimization of fitness computation process can reduce the time required for fitness computation which in effect will reduce the total time for overall protein folding prediction.

When the fitness is calculated, a particular direction for traversal can be followed say, computing from either a higher to lower (H2L) sequence number or lower to higher (L2H) sequence number, rather than counting the TN twice and then dividing the total by 2. Further, if a cumulative sum for each hydrophobic residue is stored, the available Partial Fitness (PF) computation after each hydrophobic residue can be utilized for optimization of fitness computation. In this paper, we will follow the L2H approach still maintaining the generalization of approach.

3. Mathematical analysis

Let us define the term Operation Point (OP). In case of mutation OP is the lower ordered hydrophobic residue nearest (or at) the mutation point while for crossover, OP is the lower order hydrophobic residue nearest to the crossover area. For crossover example in Figure 1, residue 14 is the OP and for mutation shown in Figure 2, residue 9 is the OP. To construct a new L2H based array of partially computed cumulative sum of fitness for a new chromosome, part of the partial fitness array from lowest order residue to OP is reconstructed every time and the remaining part is copied from one of the parents and updated with the change calculated at OP. If we assume the hydrophobic residues are equally distributed among hydrophilic residues, then reconstruction time (T_R) occurs approximately on an average for 50% of the hydrophobic residue sequence length. The remaining partially computed segments are copied and modified on an average 50% of the time. The time will be referred as partial computation (T_P). Although T_P will have some overhead, it is certainly less than T_R. The overhead of the T_P is for copying the partial fitness result from one of the parent and then updating.

So, if all the residues are equally probable for being randomly selected for mutation or crossover, an improvement of nearly 50% on an average is expected. For computing sequence of length n -

T_R^n - Reconstruction time

T_P^n - Partially computation time.

So, time required for full computation of a sequence of length *n* is T_R^n .Partial computation of sequence of length *n* is equal,

$$T_P^n = T_R^{n_1} + T_P^{n_2} \quad (1)$$

where $n = n_1 + n_2$, n_1 is the operation point (OP) and $1 < n_1 < n$.

So, the measure of improvement,

$$I = 1 - \frac{T_R^{n_1} + T_P^{n_2}}{T_R^n} \quad (2)$$

Since all the residues are equally probable of being selected for mutation or crossover then on an average the length of n_1 and n_2 each equals to $\frac{n}{2}$. Thus,

$$I = 1 - \frac{T_R^{\frac{n}{2}} + T_P^{\frac{n}{2}}}{T_R^n} \quad (3)$$

As the fitness computation is linear [6],

Hence, $$\frac{T_R^{\frac{n}{2}}}{T_R^n} = 0.5 \quad (4)$$

$$I = 1 - 0.5 - \frac{T_P^{\frac{n}{2}}}{T_R^n} \quad (5)$$

Or, $I = 0.5 - \Delta_T$,

where $$\Delta_T = \frac{T_P^{\frac{n}{2}}}{T_R^n} \quad (6)$$

The value of Δ_T will be very low since $T_P^{\frac{n}{2}}$ is required in copying and updating (addition) whereas T_R^n involves searching, comparison and addition. Hence, $I < 50\%$. The value of Δ_T has been verified to be of low magnitude by the simulation results given in section 6.

The memory overhead (MO) order is approximately proportional to number of total residues in the protein. For our algorithm, if total number of hydrophobic residue is N, the memory overhead for population size (P) in GA can be given

by, MO = ***O*** (NP). For example, for typical values of P=200 and N=300 the overhead is negligible.

4. Crossover operation and partial fitness computation

In crossover operation, at least two ancestors are involved. Partial fitness can be copied from one of the ancestor depending on the location of the crossover position and on the remaining segment of the ancestor.

4.1. Utilization of partial fitness after crossover

Consider Figure 1 which shows sequence (A) having a fitness value equal to -5 and sequence (B) having a fitness value of -2. Performing crossover immediately after residue number 14 and then following with rotation, sequence (C) is obtained. The fitness of the resulting configuration is calculated and found to be equal to -9. So, resultant child after crossover with fitness value -9 is better than its parent.

Next, the focus of proposed approach is to obtain -9 with the help of parents' fitness function rather than depending on the information from sequence (C) only.

Computation of the TN in L2H order has been indicated in Figure 2 by arrows. As shown in sequence (C) of Figure 2, the part of the protein sequence crossed and taken from (B) (indicated by solid arrows) will always be retained for the H-H topological neighbour count (with L2H order), since they need not be searched for TN of lower order sequences. However, the part of sequence crossed and taken from (A) (indicated by dotted arrows) has changed and additional neighbours appear as indicated by dot-dashed curved arrows in sequence (C) of Figure 2.

4.2. Crossover example

Consider the sequence (A) of Figure 1. There are 10 hydrophobic residues. To build a list, an array of size 10 will be required. Based on the reasons given in section 3 earlier, Partial Fitness (PF) computation for sequence (A) of Figure 1 will is as follows. In the Table, R_i is the i^{th} residue and PF_i is the partial fitness at i^{th} residue.

Table 1. Partial fitness for seq. (A) of Fig. 1

R_i	1	3	6	7	9	12	14	15	18	20
PF_i	-1	-2	-2	-3	-5	-5	-5	-5	-5	-5

Fitness, F= PF_{20} = -5

Partial fitness of sequence (B) of Figure 1 is as follows.

Table 2. Partial fitness for seq. (B) of Fig. 1

R_i	1	3	6	7	9	12	14	15	18	20
PF_i	0	0	0	0	0	0	0	-2	-2	-2

So, Fitness, F= PF_{20} = -2

The crossover operation is done and the sequence (C) of Figure 1 is obtained. We now implement PF approach for computing the fitness of sequence (C) of Figure 1.

From immediately after crossover point (i.e. residue 15) till the end residue 20, the conformation remains unchanged. This is because residue 15 needs to check for TN with higher number, namely 18 and 20, and so on. Since the structure remains the same, this required information is taken from sequence (B) of Figure 1 where it was already computed.

The modified steps for efficient fitness calculation for sequence (C) of Figure 1 are as follows.

Step1: Compute PF for residues 1 to 14.

R_i	Seq.#	1	3	6	7	9	12	14
PF_i	PF	-2	-3	-3	-5	-7	-7	-7

Step 2: Compute the change at 14 as follows.

$Change_{14} = PF_{Step1,\ 14} - PF_{B,\ 14} = (-7) - 0 = -7$

Step 3: Using the available PF of residues from Table 2 for residues 15 till 20 and adding the $Change_{14}$ found in step 2 to the existing values of the residues, we get

R_i	15	18	20
PF_i	(-2)+(-7)	(-2)+(-7)	(-2)+(-7)

Step 4: The following fully completed Table 3 for sequence (C) of Figure 1 is then built.

Table 3. Efficiently computed (PF) for seq. (C) of Fig. 1

R_i	1	3	6	7	9	12	14	15	18	20
PF_i	-2	-3	-3	-5	-7	-7	-7	-9	-9	-9

5. Mutation and partial fitness improvement

In mutation operation, only one ancestor is involved. Partial fitness can be copied from the ancestor depending on the mutation point. The details are given below.

5.1. Utilization of partial fitness after mutation

The mutation operation may be considered on the same lines as crossover operation. As the first mutation step, residue 11 of sequence (A) of Figure 3 is randomly chosen for rotation. It is randomly rotated by 180° and the configuration of (B) is obtained. Fitness is computed for (B) and found = -9.

As the sequence is traversed from 1 to 20, the TN is counted only for lower number to higher number, for example indicated by arrow direction in Figure 4. Starting at 1, one TN is found for residue 1 to 6. Then at 3, (3, 6) is found so by cumulative sum, $PF_3 = PF_1 + (-1) = -2$, etc. In L2H, 1 to 6 is computed but 6 to 1 is not computed, also 3 to 6 is computed but not 6 to 3 and so on.

Consider a mutation occurring at residue number 11 as shown in Figure 4. Since we are following L2H order for fitness computation, the hydrophobic residues after 11 will certainly not encounter any change of TN and will remain the same as it was in sequence (A) of Figure 4. So the PF computation for this unchanged part can be directly obtained from its ancestor and updated without recomputing by traversing the sequence. This technique speeds up the computation and improves the overall fitness computation efficiency.

5.2. Mutation example

An example for mutation is now considered. Partial fitness (PF) for sequence (A) of Figure 3 the PF will be computed as follows.

Table 4. Partial fitness for seq. (A) of Fig. 3

R_i	1	3	6	7	9	12	14	15	18	20
PF_i	-1	-2	-2	-2	-2	-2	-2	-4	-4	-4

So, Fitness, $F = PF_{20} = -4$

The mutation point residue number 11 is randomly chosen. For the hydrophobic residues from 1 to 9, recalculation is required and for hydrophobic residues from 12 to 20, the change can be merely copied and updated. The steps are given below.

Step1: Compute PF for residues from 1 to 9

R_i	Seq.#	1	3	6	7	9
PF_i	PF	-2	-3	-3	-5	-7

Step 2: Compute the change at 9 as
$Change_9 = PF_{Step1,\,9} - PF_{A,\,9} = (-7) - (-2) = -5$

Step 3: Using the PF of residues 12 till 20 (part of Table 4) and adding the $Change_9$ found in step 2 to the existing values of the residues, we get

R_i	12	14	15	18	20
PF_i	(-2)+(-5)	(-2)+(-5)	(-4)+(-5)	(-4)+(-5)	(-4)+(-5)

Step 4: The Final Table 5 for sequence (B) of Figure 3 is then developed.

Table 5. Efficiently computed PF values seq. (B) of Fig. 3

R_i	1	3	6	7	9	12	14	15	18	20
PF_i	-2	-3	-3	-5	-7	-7	-7	-9	-9	-9

6. Simulation results

We construct the GA on HP model for protein folding prediction and measure the time for both the full fitness computation and the partial fitness computation strategies. The partial fitness computation is done by the help of additional memory (array). We performed simulations for amino acid chain length from 20 to 335. For the purpose of this analysis, every chain was randomly filled with hydrophobic residues and hydrophilic residues. The result is shown in Figure 5. The data has been plotted from length 20 to 335. Protein length above 150 is observed to be correlated. Using curve fitting, it is found that a polynomial of order one fits the data in the best manner. The values for slope and constant are found to be 0.0125 and 41.5145 respectively giving the polynomial equation as $y = 0.0125x + 41.5145$ Using typical value of protein length x = 540, the predicted improvement is 48.26%. The improvement I is approximately 48% from experimental result. Hence from equation (6), $\Delta_T = 0.5\text{-}I \approx 0.5\text{-}0.48 = 0.02$.

7. Conclusions

In the paper, an efficient algorithm is presented to improve the computational speed of the GA. With very limited increase in memory the partial fitness is stored in cumulative manner and utilized by the descendent. In the proposed approach, full fitness computation is required only for the initial population as they will have no parents to take advantage of copying computed partial fitness. The paper presents only single point crossover and single point mutation

but the approach can be easily extended to multiple crossovers or mutations.

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

[5] 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.

[6] Santos, E.E. and Santos, E.J., "Effective and Efficient Caching in Genetic Algorithms", *International Journal on Artificial Intelligence Tools,* © Worlds Scientific Publishing Company, 2000.

[7] 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.

[8] Toma, L. and Toma, S., "Contact interactions method: A new algorithm for protein folding simulations", *Protein Science,* 1996, 5:147-153.

[9] Unger, R. and Moult J., "Genetic Algorithm for Protein Folding Simulations", *J. Mol. Biol.,* 1993, 231, 75-81.

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

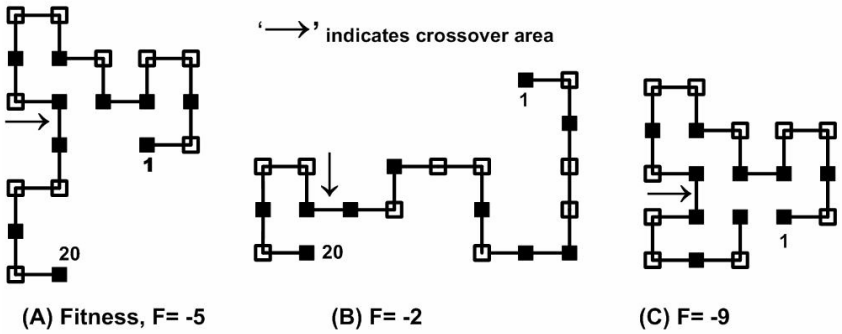

Figure 1. An example of crossover application. Black-square indicates hydrophobic residue and white-square indicates hydrophilic residue. Pairs of structures are randomly cut and pasted with the cut point randomly chosen after residue 14. The first 14 residues of (A) are joined with the least 6 residues of (B). A randomly chosen 270° rotation is applied at the joint to achieve the compact structure in (C), where fitness = -9.

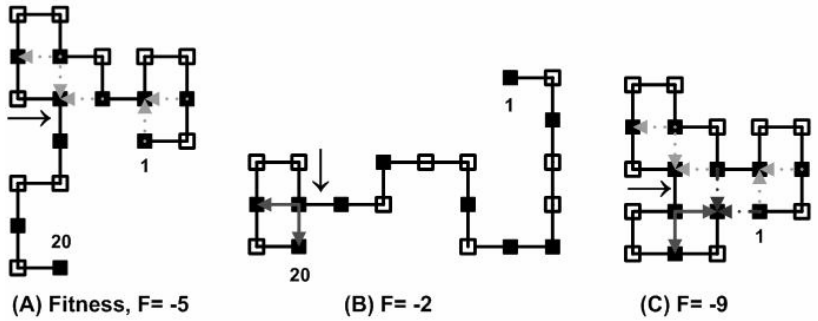

Figure 2. TN before and after crossover. TNs are indicated by dotted arrow in (A), solid arrow in (B). As a result of crossover in (C), newly achieved TNs are indicated by dot-dashed arrows.

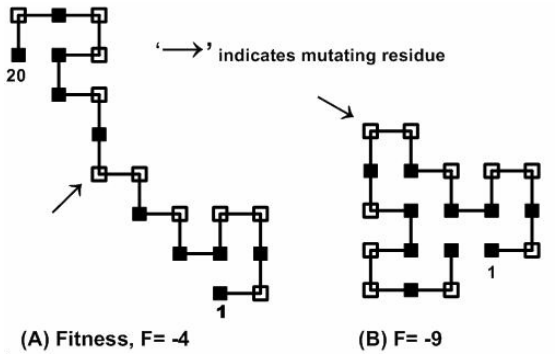

Figure 3. An example of mutation operation. Residue number 11 is chosen randomly as the pivot for move. A 180° rotation brings the structure in (A) with fitness value of -4 to structure with fitness value of -9 in (B).

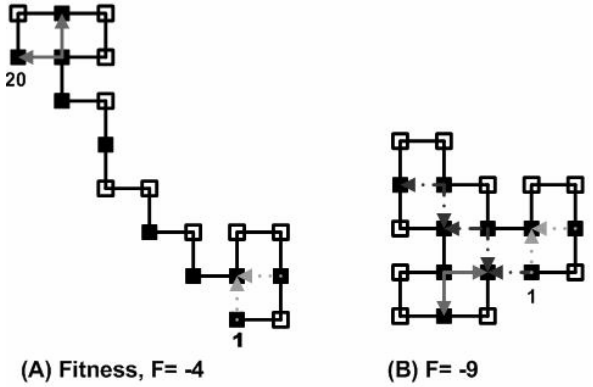

Figure 4. TN before and after mutation. TN has been indicated by dotted arrow in (A) from sequence number 1 to mutation point and for the remainder of the part these are indicated by solid arrows. Resultant (B) after mutation has new TN indicated by dot-dashed arrows.

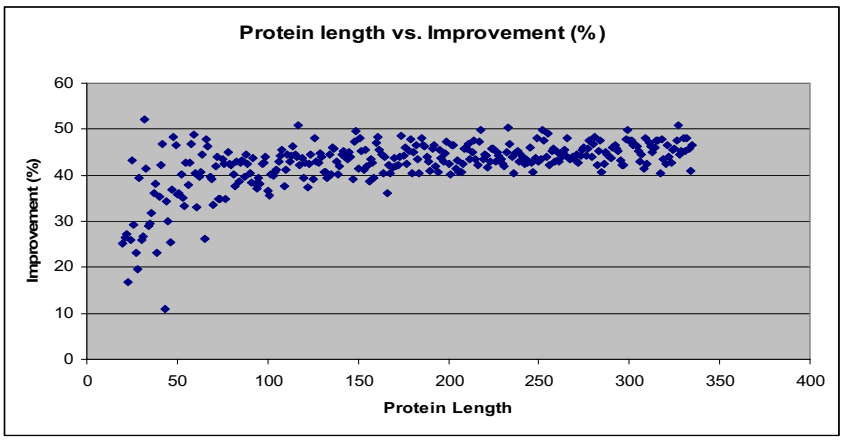

Figure 5. Protein length vs. improvement graph.