Clustered Memetic Algorithm for Protein Structure Prediction

Md. Kamrul Islam, *Student Member, IEEE* and Madhu Chetty, *Senior Member, IEEE*

***Abstract*— Memetic algorithm (MA) often perform better than other evolutionary algorithm due to their combining the local search with the process of global optimization. However, like any other evolutionary algorithm (EA), MA due to the problem of genetic drift often result in sub-optimal solutions. The problem is more aggravated when EAs are applied to search complex landscape of NP complete problem like protein structure prediction. In this paper, to help mitigate the problem of genetic drift and also to cover large search space, we propose a novel initial population generation process and a novel MA which applies clusters for seeding the initial population. Apart from reducing the impact of genetic drift, the proposed MA also avoids processing of unnecessary individuals in the population, thus significantly reducing the computational burden, especially for large protein sequences. Simulation results presented using the 2D lattice HP model show the superiority of the proposed algorithm.**

I. Introduction

Due to an astronomically large number of possible protein structures for a corresponding primary sequence of amino acids, prediction for optimal protein structure is usually performed by applying evolutionary algorithms (EAs). Although most of the EAs perform well for a global search, they are found to be inefficient in exploiting the surrounding search space leading to sub-optimal solutions [1]. In contrast, EAs in combination with local optimization procedures have been shown to improve their precision [1], [2]. In this context, local improvement procedure or local search is analogous to the learning that occurs during the lifetime of an individual. However, another way in which learning (i.e. local search) and evolution can interact is instead of coding the improvements back onto the individual, the improvement can be transferred to other individuals which Dawkins coined as meme [3]. Memetic algorithm (MA) incorporates this concept for local improvement [4] capturing the domain knowledge of individuals and passing it to the next generations. Research [5] has shown that evolutionary searches based on memes are more effective than those applying only genes for evolution. MA is a class of stochastic global search techniques which combine domain knowledge based local search heuristics and multi-agent systems within the framework of EAs [6]. MA does not have any core or clear architecture, but rather a flexible local search architecture that can be adapted for different problem domains [4]. For this reason, MA has been successful for a wide range of optimization problems such as combinatorial, continuous, dynamic, and multi-objective, etc [4].

Because of genetic drift, the MA like any other EA, suffers from the problem of population shift towards one of the possible solution peaks during optimization using EAs for a multimodal function with several peaks. Generation of a large initial population [2], [7] may not always be feasible because the evolution of a large number of individuals increases computational cost.

In order to overcome this limitation, in this paper, we propose a novel technique of clustering to extend our novel memetic algorithm (NMA) [8] which will increase the optimization search space and keep the computational cost down. The performance of the proposed clustered memetic algorithm architecture is tested with the aid of a NP-complete bioinformatics problem, protein structure prediction (PSP) for 2D hydrophobic-hydrophilic (HP) lattice model [9].We have introduced a novel *dynamic individual generation* (DIG) technique for HP model protein here and have showed that DIG guarantees a *Self Avoiding Walk* (SAW) and in addition, the technique is much faster than the traditional approach.

Several evolutionary algorithms including ant colony optimization (ACO), tabu search (TS), self organizing map (SOM), progressive-based computing approaches (e.g. chain growth (CG), pruning enrichment Rosenbluth method (PERM)) have been used to solve protein structure prediction (PSP) problem and a detailed review of those algorithms can be found in [10]. Recent literature shows that immune algorithm (IM) [11] and estimation of distribution algorithm (EDA) [12] have also been used to solve protein structure prediction on an HP model. Though memetic algorithm shows success in different problem domains, less effort has been made to solve PSP using memetic algorithm. The rest of the paper is organized as follows: section II describes new initial random population generation technique; section III explains our novel clustering technique; empirical results are shown in section IV and ends with a short conclusion outlining the summary of this paper and future work directions.

II. Population Initialization

In EAs, an initial population of randomly generated candidate solutions comprises first generation. An individual, $\boldsymbol{x}$, is a sequence of genes and each gene takes value randomly from a set of possible values. If $l(\boldsymbol{x})$ is the length of the gene sequence of individual $\boldsymbol{x}$ then an individual with a binary gene string can be defined as in eqn. 1

$$\boldsymbol{x} = x_1 x_2 x_3 \ldots x_{l(\boldsymbol{x})} \quad \text{where} \quad \forall_i x_i \in \{0, 1\} \qquad (1)$$

For PSP problem, initial individual generation is not as straight forward as in many other applications because of the requirement of *Self Avoiding Walk* or SAW so that no

Md. Kamrul Islam is with the Gippsland School of Information Technology, Monash University, Gippsland Campus, Churchill 3842, Australia (phone: +61 351 226135; email: Kamrul.Islam@infotech.monash.edu.au).

Madhu Chetty is with the Gippsland School of Information Technology, Monash University, Gippsland Campus, Churchill 3842, Australia (phone: +61 351 227148; email: Madhu.Chetty@infotech.monash.edu.au).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

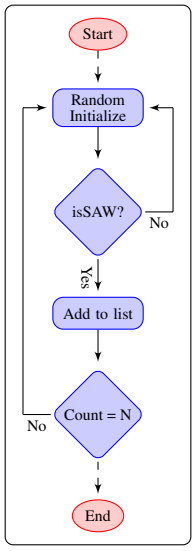


Fig. 1. Flow-Diagram of General Random Initialization For PSP

two residues cross each other. An individual protein is said to be SAW iff two residues, $\mathbf{r_i}$ and $\mathbf{r_j}$, where $i \neq j$ do not have the same position in the lattice.

A. Traditional Method

Traditionally, an individual in a 2D HP model is generated using random moves (genes) that could not guarantee a SAW individual. This necessitates a SAW check process for discarding non-SAW individuals and also the generation of that individual again. This process is repeated until the necessary number of individuals is generated. The flow diagram of this process is shown in Fig. 1.

Considering 2D relative encoding [13] involves three different possible moves, namely, *F-Forward*; *L-Left*; *R-Right* as shown in Fig 2(a). From *eqn. 1*, the conformation moves of an individual with relative encoding in 2D square lattice can be defined as *eqn. 2*.

$$\boldsymbol{x_{C_i}} = x_1 x_2 x_3 \ldots x_{(l-2)} \quad \text{where} \quad \forall_i x_i \in \{F, L, R\}. \quad (2)$$

Since the first move of a conformation is always a forward (F) move [13], a protein sequence of l, there can be $(l-2)$ possible moves in any conformation. This results in a large number (i.e. $N_p = 3^{(l-2)}$) of possible conformations some of which are invalid due to SAW requirement.

As this process involves in random selection of moves, SAW individual generation using this process takes an exponentially long time. Considering a sequence of length more than 4 residues can have non-SAW individuals, we note that a sequence of 5 residues can have two non-SAW individuals and the number of non-SAW individuals increases exponentially as the number of residues in a sequence increases. If the length of a non-SAW conformation is increased, it continues to remain non-SAW irrespective of what the next moves. Hence, in the worst case, there can be an exponentially large, $N_w = \left(2 \times 3^{(l-5)}\right)$, non-SAW conformations. This is because, the worst case appears when a loop is formed at the beginning and continues to grow. A confirmation of 5 residues can have two non-SAW individuals by forming loops (in relative encoding *FLLL* and *FRRR*). If moves are increased of these two non-SAW conformations they will remain non-SAW. As in relative 2D encoding there are 3 possible moves at any point thus taking into consideration a protein of length l there can be 3^{l-5} possible moves for rest of the resides of these two conformations. Assuming the probability of getting a non-SAW for a conformation with the worst case being P_{wt}, for a protein of length l, P_{wt} is calculated as shown by *eqn.* 3.

$$P_{wt} = \frac{N_w}{N_p} = \frac{2 \times 3^{(l-5)}}{3^{(l-2)}} = \frac{2}{3^3} = 0.074 \quad (3)$$

We note that, the worst case probability of forming a non-SAW is fixed irrespective of the length of the protein.

B. Proposed Method

In the proposed method, *dynamic individual generation* (DIG), conformation is generated randomly but by using 2D points of the lattice and absolute moves together instead of relative moves alone as in the traditional method. The process starts from the mid point of the lattice model. The process begins with a *possible move set*, S_{P_i}, {E(east), W(west), N(north), S(south)} for each residue i, and is updated according to possible moves of the residue. Then before taking any random move for i form S_{P_i}, we verify whether the position in the lattice is previously occupied by another residue or not. If the position is empty then the move is implemented for the residue and the point in the lattice is updated accordingly. If not, then the move is discarded from the S_{P_i} and again a random move is chosen from the truncated S_{P_i}. If S_{P_i} becomes empty, then the process, recognizing that the previous move was a wrong move passes the control back to the previous residue, $(i-1)$. A new move, ignoring the earlier move, is taken from the truncated move set $S_{P_{i-1}}$. This process is continued till all moves are generated for the individual.

Algorithm 1 Dynamic Individual Generation (Part 1)

```
1: procedure INITIALIZATION                          ▷ Random initialization
2:     indvCount ← 0
3:     while indvCount < PopulationSize do           ▷ Untill all the individual generated
4:         startPoint = latticeCenter
5:         PointsList.add(startPoint)
6:         UpdateLattice(startPoint)
7:         secondPoint = eastNeighbour(startPoint)
8:         PointsList.add(secondPoint)
9:         UpdateLattice(secondPoint)
10:        GetPointsMinusTwo(2, E, secondPoint)▷ 2 denotes two points have been generated
11:    end while
12: end procedure
```

The algorithm for the implementation of the proposed method is shown in two parts: Algorithm-1 and Algorithm-2. In Algorithm-1, the computation of first two points, which are fixed for any conformation, is carried out. In Algorithm-2, the recursive generation of rest of the points is performed.

The worst case scenario causing maximum back tracking occurs only with a spiral conformation. The probability of spiral conformations reduces exponentially with the increase in sequence length. There is exactly one way to form this and if the length of the spiral in is S_l then the probability of forming this spiral in is $\frac{1}{3}^{(S_l-2)}$. This is because, from each residue i to the probability of choosing the following point of the residue $(i+1)$ is $\frac{1}{3}$ because it can go back to the previous residue $(i-1)$. The proposed method guaranties that it will come out of the spiral conformation after visiting all the possible moves inside and once it comes out of the spiral in conformation it will never revisit that path again. So our proposed method of individual generation is complete as it always guaranties termination with a SAW individual. As

Algorithm 2 Dynamic Individual Generation (Part 2)

```
 1: procedure GETPOINTSMINUSTWO (indx, prvMove, prvPoint)   ▷ Generates Rest of the Points
 2:     if index = ProteinLength then
 3:         return true
 4:     else
 5:         PossMoveSet= ConstructPossMovesSet(prvMove)
 6:         NonPossMoveSet={}
 7:         while true do
 8:             curMove=GetRandMove(PossMoveSet,NonPossMoveSet)
 9:             if !isOccupied(curMove, prevPoint) then
10:                 PointsList.add(TakeTheMove(curMove,prvPoint))
11:                 UpdateLattice(PointList[indx])
12:                 if GetPointsMinusTwo(indx+1,curMove,PointList[indx]) then
13:                     return true
14:                 else
15:                     UndoLattice(PointList[indx])
16:                     PointList.remove(indx)
17:                     NonPossibleMoveSet.add(curMove)
18:                     if PossMoveSet.Count=NonPossMoveSet.Count then
19:                         return false
20:                     end if
21:                 end if
22:             else
23:                 NonPossibleMoveSet.add(curMove)
24:                 if PossMoveSet.Count=NonPossMoveSet.Count then
25:                     return false
26:                 end if
27:             end if
28:         end while
29:     end if
30: end procedure
```

we are following relative encoding, the algorithm kicks off by fixing the first two points for first move as a *forward (F)* move. Next, for any residue i, the conformation has three possible neighbor points for the following $(i+1)^{th}$ residue. To keep track of allowed moves at any point i, we maintain two sets: S_{P_i} (Set of all *Possible* moves) and S_{NP_i} (Set of *Not-Possible* moves).

As shown in (Algorithm. 2) for any i^{th} residue, S_{P_i} is first initialized (line 5-6) with all permissible moves and S_{NP_i} as an empty set. A free location is chosen from S_{P_i} in a random manner (line 8). If the point is already occupied by another residue, the random move becomes invalid and gets included in the set S_{NP_i} (line 23). Another random move is chosen from $\{S_{P_i} - S_{NP_i}\}$. If $S_{P_i} = S_{NP_i}$ (line 24), this implies that the move taken at $(i-1)$ was invalid. Hence, we undo the move (line 15-16) and is added into the S_{NP_i} set (line 17) of $(i-1)^{th}$ residue. The control returns to the previous position (line 15) in a recursive manner returning *false* (line 25). A new move is again chosen randomly for $(i-1)$ (line 8). This process continues until all valid points are found for the sequence (line 2) whereupon it terminates. So the algorithm will always terminate with a SAW conformation and still it is a random process.

Lemma 1. *The DIG guarantees Self Avoiding Walk and a valid conformation .*

Proof. S_{NP_i} keeps track of the moves that are not permissible at any point i. This ensures it will not to follow same path again and again. A residue takes a point in the lattice only if the point is not already occupied by another residue thus this ensures the generation of a SAW conformation all the time. Furthermore, the number of total possible conformations is more than the number of non-SAW conformations conformation which essentially confirms that the proposed process will always terminate with a SAW conformation. □

Lemma 2. *In a DIG, the probability of a worst case scenario (upper bound) reduces exponentially with sequence length.*

Proof. As stated earlier, in proposed the DIG method, the worst case scenario occurs only when the conformation forms a spiral in as shown in Fig. 2(b), 2(c) and 2(d). If the 2D rectangular spiral in type conformation has a dimension of $a \times b$ then the number of residues on the boundary will be as shown in *eqn.* 4.

$$\begin{aligned} 2a + 2(b-2) &= 2a + 2b - 4 \\ &= 4(a-1) \text{ when } a = b, \text{ for square} \end{aligned} \tag{4}$$

The number of points *inside* the boundary will be equal to *eqn.* 5.

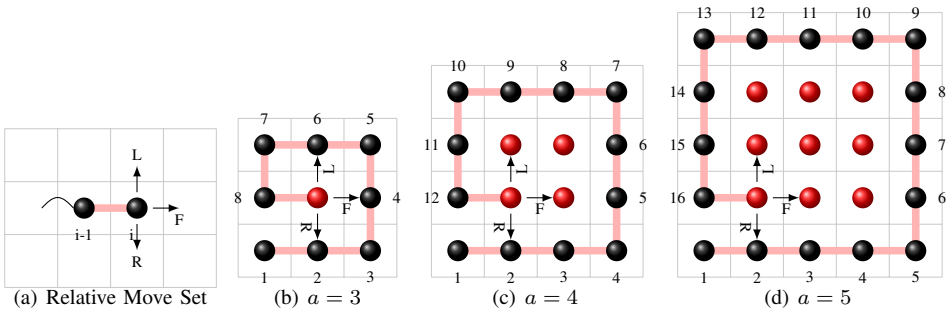
$$\begin{aligned} (a-2) \times (b-2) &= ab - 2a - 2b + 4 \\ &= (a-2)^2 \text{ when } a = b, \text{ for square} \end{aligned} \tag{5}$$

In case of a square spiral there will be the maximum number of points inside the boundary. Considering the case of a spiral for a square of length a. By adding *eqn.* 4 and *eqn.* 5 we find *eqn.* 6 as the total number of points in the square spiral.

$$a^2 = 4(a-1) + (a-2)^2 \tag{6}$$

The *eqn.* 6 shows that the number of points inside the boundary is polynomial whereas the number of points on the boundary is linear. If a conformation results in a spiral, then it will search all the possible combinations inside. If the number of residues needing a position inside the spiral is more than the number of available points inside, it results in maximum backtracking. Eventually, it will comeback through backtracking according to the proposed algorithm but that will be the worst case scenario for this algorithm.

The *eqn.* 7 when the length of the square is greater or equal 9, the number of points inside will be greater than the number of points in the boundary. That also means that there must be at least $(9^2 + 1)$ to get the worst case spiral in where the



(a) Relative Move Set (b) $a = 3$ (c) $a = 4$ (d) $a = 5$

Fig. 2. (a) shows relative move sets. The worst case scenario for different values of a are shown in (b), (c) and (d)

number of points needed to back track is greater than the number of points not needed to back track.

$$\begin{aligned} (a-2)^2 - 4(a-1) &> 0 \\ a^2 - 8a - 8 &> 0 \\ (a-4)^2 &> 8 \\ a &> 6\sqrt{2} \text{ or } a > 8.49 \end{aligned} \tag{7}$$

To get a *spiral* for a square of length a at least $4(a-1)+1$ points (one more that the number of points form the boundary) are needed. So, the required number of moves is $4(a-1)-1$ and $3^{4(a-1)-1}$ different conformations in relative encoding. From this many only 2 will form a *spiral* for a 2D square lattice. The probability of reaching at this *spiral* condition is shown in *eqn.* 8 taht is the worst case probability of the proposed method, P_{wp}.

$$P_{wp} = \frac{2}{3^{4(a-1)-1}} = 2 \times \left(\frac{1}{3}\right)^{4(a-1)-1} = 2 \times \left(\frac{1}{3}\right)^{4a-5} \tag{8}$$

So, the probability of forming a *spiral* is exponentially reduces with the length of the square a, and in turn, with the length of the sequence, S_l as $a \propto S_l$.

In the traditional approach, it can happen that a loop has been formed at any of the stage with a minimum of 4 moves (*FLLL* or *FRRR*) but it continues to build the rest of the conformation. The probability of getting this type of conformation is $2 \times \left(\frac{1}{3}\right)^4$ and in the worst case this can form at the beginning of the conformation and the probability of that is $2 \times \left(\frac{1}{3}\right)^3$. So it is fixed and much higher than the proposed approach. □

Following part of this paper elaborates the novel clustering technique.

III. Clustering NMA

A simple genetic algorithm (GA), when applied to a multi-modal function, converge to only one of the peaks. Moreover this peak is randomly chosen due to the well-known *genetic drift*: the simple GA has no means of deciding amongst the different global peaks, and only the stochastic variations due to the genetic operators can make the population drift to one of these peaks [14]. As a result, convergence may be very slow until the population drifts to one of the basins or, even worse, the algorithm may get stuck in a local optimum.

Memes can preserve domain knowledge of peaks efficiently and it can also transfer the knowledge to other individuals thus enabling the EA to explore and exploit the entire search space effectively, thereby, preventing the occurrence of local optima. However, since genetic drift drives all individuals to one peak, a single set of population will fail to identify memes corresponding to different peaks of the search space. Therefore making it necessary to explore and exploit the entire search space to get rid of local optima. Preserving the domain information of each of the peaks in EAs will help to explore and exploit the entire search space effectively. Memes can preserve domain knowledge of peaks efficiently and can also transfer the knowledge to other individuals. The literature [15] shows that clustering data set helps to segregate global optima effectively and efficiently but a set of limited individuals cannot cover all the possible peaks. Considering large initial population to cover all possible peaks may not always be feasible due to large computational time [2], [7]. To overcome these problems, we propose a novel initial data set clustering technique to help in both exploring various peaks and covering as many individuals as possible efficiently.

The proposed method for clustering works in two stages. In stage 1, the number of clusters, say κ, are formed for a new population set. Each cluster evolves independently using NMA [8] for a specified number of iterations, I_κ , thereby generating a set of best individuals in each cluster. The substructures (memes) present in the best individuals carry the base information of the peak to which the solution is converging. These memes from each cluster are then used to generate new individuals resulting in one main cluster of population. Next, in stage 2,we generate N_κ individuals from the memes of each cluster. If N is the total number of individuals then $N_\kappa = \frac{N}{\kappa}$. After that, the newly generated individuals from memes that will evolve in the main cluster using NMA [8] with regular crossover, mutation and using a set of improved local search techniques. The overall architecture of the proposed clustering NMA is shown in Fig. 3.

The meme generation technique with the entire mechanism of transferring domain knowledge through the memes and the resulting improved local search techniques are described in subsequent sections.

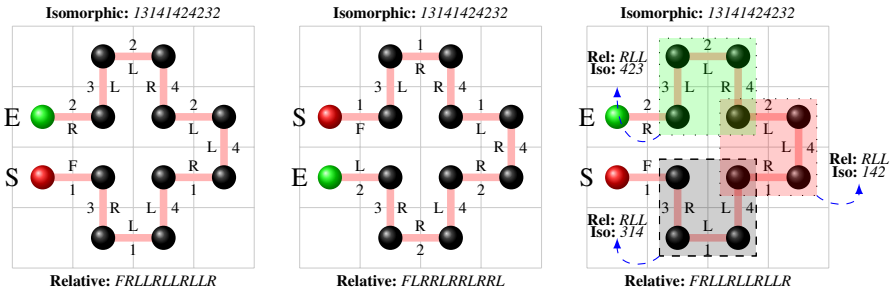


Fig. 4. Fig. (a) and Fig. (b) show same conformation in Relative and Isomorphic encoding. Comparison of memes in relative encoding and isomorphic encoding are shown in Fig. (c). Here **S** indicates *Start* point and **E** indicates *End* point.

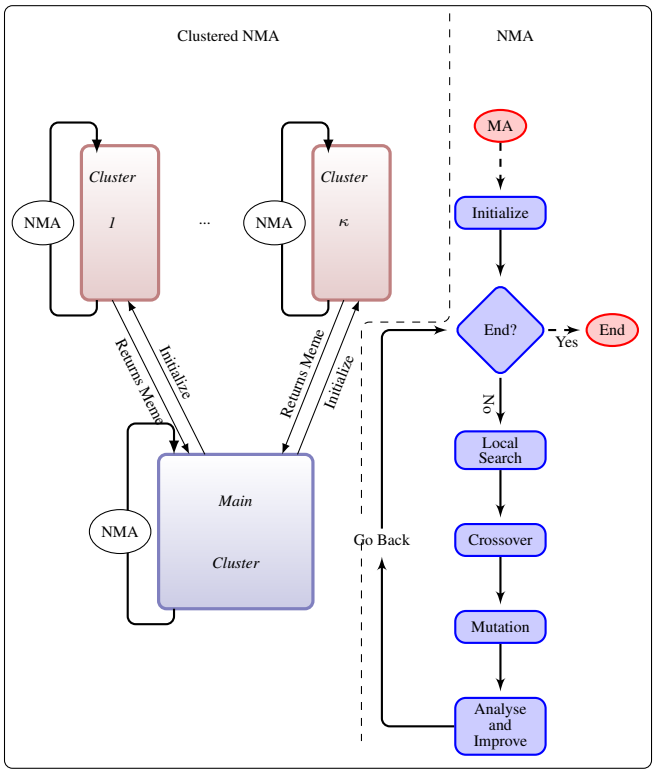


Fig. 3. Clustered Novel Memetic Algorithm (CMA)

A. Meme Generation

To generate meme in each cluster, we use our schema preservation technique reported earlier [8]. If we set the *cut off* value, χ, to a higher value (i.e. greater than 0.8) which means that same move occurred at the same position of the best individual more that 80% times, then the schema generated by this method will be the base substructure of the best individuals of the region the solution converges and we can call it a *meme*. This technique identifies the substructures that remain fixed in the best individuals. We modify the schema preservation technique when creating a list of best individuals. Rather than using relative encoding for comparison between different individuals isomorphic encoding [13] is used whereas memes are kept in relative encoding. Isomorphic encoding is a better for differentiating between two individuals than relative encoding. For example, lets consider two conformations in relative encoding *FRLLRLLRLLR* and *FLRRLRRLRRL* as shown in Fig. 4(a) and Fig. 4(b). Though the relative encodings are different, the conformations are basically the same, that is, a reflection of each other. The isomorphic encodings for these two conformations are same. So, comparing using isomorphic encoding will ensure that these types of identical conformations do not appear in the list.

B. Local Improvement

Local improvement is done using meme replacement, pull move and mutation operation, adaptively within the tabu search framework. In tabu search, tabu list ensures the same individual is not generated earlier.

1) Meme Replacement: In main cluster, meme replacement technique is used as a local improvement. Meme structure is a tuple containing the substructure in relative encoding, the start point and the end point. All memes generated from different clusters are kept in a list and in the meme replacement method each meme is used as a candidate for improvement. A reflection of each meme is also used during the meme replacement that ensures no information is lost due to reflection of the conformation. Although isomorphic encoding is better for ensuring diversity, relative encoding is better at representing a substructure than a isomorphic encoding because representation of substructure in isomorphic encoding varies with previous moves whereas in relative encoding it does not vary with previous moves, rather it depends on the current move. This means, a substructure with relative encoding can be placed as it is and does not need any conversion. On the other hand, if memes were kept using isomorphic encoding then they could not be placed as it is rather they would need a conversion mechanism for meme replacement. As an example, in Fig. 4(c) the substructure in dashed rectangle has three occurrences and all of them have relative encoding *RLL* whereas the isomorphic encoding are different; *314*, *142* and *423*.

2) Pull Move: Local improvement by pull move, introduced by Lesh *et al* [16] which was subsequently extended by

[17], [18], [19], is very effective in exploiting the neighbour. On top of this we propose some related improvements on pull moves which ensures all possible neighbour exploitation. There are twelve possible scenarios for a pull move in a 2D lattice. We have defined a conformation in a 2D lattice and redefined a possible pull move scenario for a 2D lattice.

Definition III.1. (Conformation in a 2D Square Lattice) A conformation in a 2D Square Lattice, $C_{2\mathcal{L}}$, as an ordered pair $C_{2\mathcal{L}} = (V, E)$ where,
$V = \{A_1, A_2, \ldots, A_N\}$ where N is the number of amino acids and
$E = \forall_{1<i<N}\{(A_{i-1}, A_i) \cup (A_i, A_{i+1})\}$.
Here, V is the set of **vertices** of *amino acids* and E is the set of **lines** connecting the *amino acids* where the length of each line can be defined as $\forall_{0<j<N}|(A_j, A_{j+1})| = 1$ and each vertex of amino acid can be defined as $\forall_{1\leq i\leq N}\{A_i = (x_i, y_i)\}$.

Definition III.2. (Redefined Pull Move) A pull move is possible iff starting at a vertex (*amino acid*) A_i where $1 \leq i < N$, there is an empty space on either sides of A_{i+1} in a 2D square lattice and there is an empty space in the corresponding side of A_i or it is occupied by A_{i-1}.

According to definition III.2 twelve (12) different scenarios of pull move may occur as we advance through the chain of a conformation, $C_{2\mathcal{L}}$ as shown in the Fig. 5. Formally, we define the direction of $\overrightarrow{A_i, A_{i+1}}$ as $\overrightarrow{d}$ and the two sides of the direction, $\overrightarrow{d}$, on a 2D plane as a positive(+) and a negative (−) side. We also define positive(+) side of A_{i+1} as **C** and negative(−) side of A_{i+1} as **Ć** and likewise positive(+) side of A_i as **L** and negative(−) side of A_i as **Ĺ**. Then if we get an empty space at **C** and either **L** is empty or occupied by A_{i-1} then a pull move is possible at the positive side of $\overrightarrow{d}$. The same orientation can occur for **Ć** & **Ĺ** in the negative side of $\overrightarrow{d}$ which will create a possibility of pull move on the negative side of $\overrightarrow{d}$. More interestingly, both of these two scenarios can occur at the same time which will create room for pull moves on both the sides. This makes three different scopes for pull move in a direction and there can be four such directions as shown in the Fig. 5 which will generate twelve possible pull move scenarios in total.

Definition III.3. (Reverse Pull Move) A reverse pull move is possible iff starting at a vertex (*amino acid*) A_i where $1 < i \leq N$, there is an empty space on either sides of A_{i-1} in a 2D square lattice and there is an empty space on the corresponding side of A_i or it is occupied by A_{i+1}.

According to definition III.3 there are twelve(12) different scenarios of pull move that may occur as we advance through the chain of a conformation, $C_{2\mathcal{L}}$ in reverse direction as shown in the Fig. 6. Formally, if we define the direction of $\overrightarrow{A_i, A_{i-1}}$ as $\overrightarrow{d'}$ then the two sides of the direction, $\overrightarrow{d'}$, on a 2D plane can be defined as a positive(+) and a negative(−) side. We define positive(+) side of A_{i-1} as **C** and negative(−) side of A_{i-1} as **Ć** and likewise positive(+) side of A_i as **L** and negative(−) side of A_i as **Ĺ**. If we get an empty space at **C** and either **L** is empty or occupied by A_{i+1} then there is a possible pull move at positive side of $\overrightarrow{d'}$. The same situation can occur for **Ć** & **Ĺ** in the negative side of $\overrightarrow{d'}$ which will generate a possible pull move situation on the negative side of $\overrightarrow{d'}$. So, same as pull move at the forward direction, there can be twelve different possibilities of pull move that can occur in reverse direction as well.

With this reverse pull move it is possible to exploit possible neigbours that would not be possible with only the general pull move. For instance, Fig. 7 shows the difference between general pull move and the proposed reverse pull move. Residue A_{k+4} comes to the position of **L** in general pull move and other residues A_j where $j < k+4$ are updated as per the definition of pull move [16]. Whereas in reverse pull move A_{k+6} comes to the position of **L** and other residues A_j where $j > k+6$ are updated as per the definition of pull move.

3) Mutation: Local improvement by mutation is another way to exploit the neighbour of a conformation $C_{2\mathcal{L}}$. To make the mutation process effective, we set a mutation pivot first using definition III.4. Subsequently we performed a two point mutation on it based on mutation pivot. We have found that a two point mutation is more effective than a single point mutation.

Definition III.4. (Mutation Pivot) A vertex (*amino acid*) $A_i = (x_i, y_i)$ where $1 < i < N$ is a mutation pivot iff there is an empty cell in its four neighbours. Formally, if there is an empty cell among $\{(x_i, y_i+1), (x_i, y_i-1), (x_i+1, y_i), (x_i-1, y_i)\}$.

TABLE I
BENCHMARK SEQUENCES, HERE E* SHOWS THEIR OPTIMUM FITNESS

Inst.	Len.	Sequence	E*	Ref
B1	48	2ph2p2h2p2h5p10h6p2(2h2p)h2p5p	-23	[13]
B2	50	2h3(ph)p4hp2(h3p)h4p2(h3p)hp4h3 (ph)p2h	-21	[13]
B3	60	2p3hp8h3p10hph3p12h4p6hp2hphp	-36	[13]
B4	64	12h2(ph)2(2p2h)2ph2p2(2h2p)2(hpph) hp2(ph)p12h	-42	[13]
B5	85	4h4p12h6p12h3p12h3p12h3ph2p2h2p2h2 phph	-53	[13]
B6	100a	3p2h2p4h2p3h2(phh)p4h8p6h2p6h9php 2hp11h2p3hp2hph2php3h6p3h	-50	[13]
B7	100b	6php2h5p3hp5hp2h4p2h2p2hp5hp10hp2h p7h11p7h2php3h6php2h	-48	[13]

IV. EXPERIMENTAL RESULT

Experiments are carried based on the benchmark sequences shown in Table I. Only challenging benchmark sequences are considered here as comparisons for shorter length sequences have already been presented using NMA in [8]. Performance analysis between the proposed initial population generation algorithm and the traditional population generation algorithm is shown in section IV-A. Outputs of Clustered NMA for different sequence of protein are shown in section IV-B.

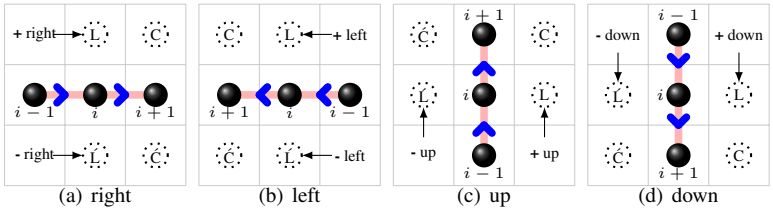


Fig. 5. Different possible scenarios for Pull Move in 2D square lattice are shown in (a), (b) , (c) and (d).

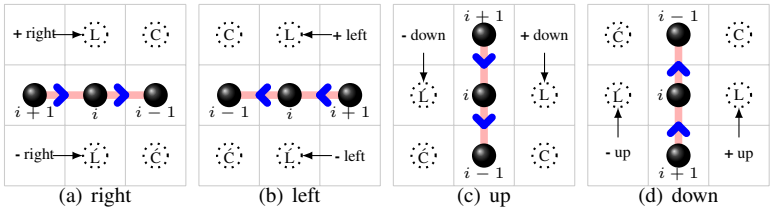


Fig. 6. Different possible scenarios for Reverse Pull Move in 2D square lattice are shown in (b), (a) , (c) and (d).

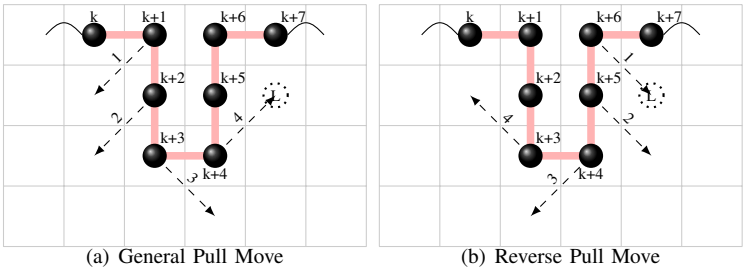


Fig. 7. Difference between General Pull Move(in Fig. (a)) and Reverse Pull Move(in Fig. (b)) are shown in this figure.

TABLE II
TIME (IN SECOND) NEEDED TO GENERATE 200 INDIVIDUALS OF DIFFERENT LENGTH SEQUENCES

	Seq.	Run1	Run2	Run3	Run4	Avg
Proposed	B2	0.093	0.078	0.062	0.078	0.077
	B3	0.109	0.093	0.093	0.093	0.097
	B4	0.124	0.124	0.140	0.109	0.124
	B5	0.249	0.187	1.265	0.156	0.464
	B6	0.234	0.296	0.249	0.234	0.253
	B7	1.374	0.874	0.218	12.640	3.776
Traditional	B2	1.609	1.546	1.671	1.718	1.636
	B3	5.843	6.171	6.234	6.593	6.210
	B4	11.343	10.109	11.515	10.109	10.769
	B5	191.123	167.123	191.983	148.264	174.623
	B6	1291.429	1139.570	1104.586	1031.790	1141.844
	B7	1038.524	1044.368	1027.946	971.087	1020.481

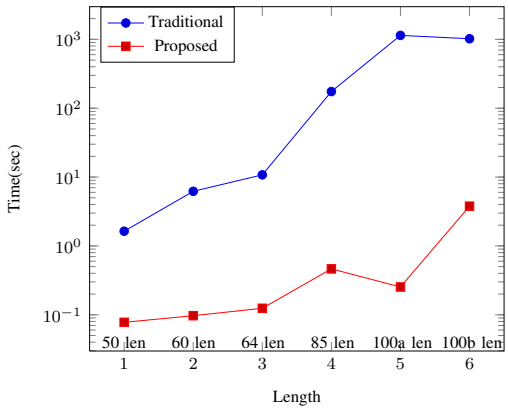


Fig. 8. Comparison of average time (in second) needed in individual generation of different length between tradition and proposed method

A. Initial Population

A comparison between traditional and proposed method of population generation is shown in Table. II. Longer length sequences (see Table I) are specifically chosen for the comparison to make the impact more conspicuous. In evolutionary algorithms besides initial random population, 10%-20% new random individuals are also needed during replacement due to elitism or other improvement techniques in every generation. Hence, if the new random individual generation takes longer time then it will affect overall performance of the whole algorithm. The difference in the average time needed for individual generation in the proposed and traditional approach is shown in the semi-log plot (Fig. 8).

B. Clustered NMA

If we can formulate memes correctly then they will preserve the base information of peaks when they are be used to generate new individuals they pass on the information. So the new individuals, therefore, should at least reproduce the same peak or a better peak than initials clusters in the course of evolution. Based on this hypothesis, we have set the initial cluster size $\kappa = 4$ and each of the clusters generates $N = 200$ individuals. Every cluster is evolved $I_\kappa = 50$ for iterations

TABLE III

RESUT FOR CLUSTERED NMA (NUMBER OF ITERATION NEEDED TO REACH THE FITNESS IS SHOWN IN BRACKET)

Seq	Run	**Cluster - 1**	**Cluster - 2**	**Cluster - 3**	**Cluster - 4**	**Final**
	1	-21	-21	-22	-21	-22(22)
B1	**2**	-22	-22	-22	23(27)	
	3	-21	-22	-21	-22	-22(1)
	1	-21(41)				
B2	**2**	-21(25)				
	3	-21(23)				
	1	-36(27)				
B3	**2**	-35	-35	-35	-34	-35(1)
	3	-34	-35	-34	-34	-35(1)
	1	-40	-39	-39	-39	-40(13)
B4	**2**	-36	-39	-38	-36	-40(37)
	3	-39	-42(35)			
	1	-49	-49	-49	-50	-50(23)
B5	**2**	-46	-50	-51	-49	-51(376)
	3	-48	-49	-48	-48	-50(1692)
	1	-44	-45	-44	-42	-45(1)
B6	**2**	-45	-45	-45	-45	-45(1)
	3	-45	-45	-47	-45	-47(1)
	1	-42	-44	-42	-44	-44(1)
B7	**2**	-42	-43	-44	-42	-45(259)
	3	-45	-46	-46	-45	-46(19)

TABLE IV

RESULTS ACHIEVED BY DIFFERENT SEARCH ALGORITHMS

Inst.	BestIM [11]	BestEDA [12]	BestGGA [13]	BestCMA
B1	-23	-23	-23	-23
B2	-21	-21	-21	-21
B3	-35	-35	-36	-36
B4	-42	-42	-42	-42
B5		-52		-52
B6		-48		-48
B7		-47		-46

for generating memes and then the memes are transferred from these clusters to the main cluster. The algorithm was run sequentially for different benchmark sequences as shown in (see Table I). The simulation was carried out for up to a maximum of 3 hours or earlier if the optimum was obtained. The results for different sequences are shown in Table III using clustered NMA where same fitness does not necessarily mean same structure. From the results, it is clear that main cluster is able to reproduce at least the same peak or produces a better peak than the stage 1 clusters. The best result achieved from our proposed clustered NMA (CMA) after running the algorithm for 6 hours is compared with other approaches in Table IV showing that our proposed technique produces competitive results compare to other approaches.

V. CONCLUSION

Random population generation, which is an important part of any EA including memetic algorithm, is constrained, in the case of protein structure Prediction, by the need for an individual to satisfy SAW. The proposed MA incorporating the novel DIG technique generates individuals quickly even for longer protein sequences. The clustering technique presented here establishes that when the memes are properly identified, they contain correct domain information of peaks. If this information is passed to other individuals, it reproduces better or at least the same peaks. We have shown that both meme generation technique and the meme replacement technique presented in this paper play a significant role in proposed clustered memetic algorithm. The proposed clustered novel memetic algorithm is easily scalable in a grid or distributed architecture environment.

REFERENCES

[1] C. R. Houck, J. A. Joines, M. G. Kay, and J. R. Wilson, "Empirical investigation of the benefits of partial lamarckianism," *Evolutionary Computation*, vol. 5, no. 1, 1997.

[2] A. C. Martínez-Estudillo, C. Hervás-Martínez, F. J. Martínez-Estudillo, and N. García-Pedrajas, "Hybridization of evolutionary algorithms and local search by means of a clustering method," *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 3, pp. 534–545, 2006.

[3] R. Dawkins, *The selfish gene*. Oxford University Press, 1976.

[4] N. Krasnogor, "An unorthodox introduction to memetic algorithms," *ACM SIGEVOlution*, vol. 3, no. 4, pp. 6–15, 2008.

[5] L. Bull, O. Holland, and S. Blackmore, "On meme-gene coevolution," *Artificial Life*, 2000.

[6] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts towards memetic algorithms," California Institute of Technology, Tech. Rep., 1989.

[7] H.-S. Kim and S.-B. Cho, "An efficient genetic algorithm with less fitness evaluation byclustering," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 2, 2001, pp. 887–894.

[8] M. K. Islam and M. Chetty, *Novel Memetic Algorithm for Protein Structure Prediction*, ser. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2009, ch. AI 2009: Advances in Artificial Intelligence, p. 412421.

[9] B. Berger and T. Leighton, "Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete," *Journal of Computational Biology*, vol. 5, no. 1, pp. 27–40, 1998.

[10] X. Zhao, "Advances on protein folding simulations based on the lattice hp models with natural computing," *Applied Soft Computing*, vol. 8, no. 2, pp. 1029–1040, 2008.

[11] V. Cutello, G. Nicosia, M. Pavone, and J. Timmis, "An immune algorithm for protein structure prediction on lattice models," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 101–117, 2007.

[12] R. Santana, P. Larraaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 418–438, 2008.

[13] M. T. Hoque, M. Chetty, and A. Sattar, *Genetic Algorithm inAb Initio Protein Structure Prediction Using Low Resolution Model: A Review*. Springer Berlin / Heidelberg, 2009, vol. 224, ch. Biomedical Data and Applications, pp. 317–342.

[14] C. Hocaoğlu and A. C. Sanderson, "Multimodal function optimization using minimal representation size clustering and its application to planning multipaths," *Evolutionary Computation*, vol. 5, no. 1, pp. 81–104, 1997.

[15] M. Pelikan and D. E. Goldberg, *Genetic Algorithms, Clustering, and the Breaking of Symmetry*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2000, pp. 385–394.

[16] N. Lesh, M. Mitzenmache, and S. Whitesides, "A complete and effective move set for simplified protein folding," in *Proceedings of the seventh annual international conference on Research in computational molecular biology*, 2003, pp. 188 – 195.

[17] M. T. Hoque, M. Chetty, and L. S. Dooley, *A Hybrid Genetic Algorithm for 2D FCC Hydrophobic-Hydrophilic Lattice Model to Predict Protein Folding*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, ch. AI 2006: Advances in Artificial Intelligence.

[18] ——, "A guided genetic algorithm for protein folding prediction using 3d hydrophobic-hydrophilic model," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 2339–2346.

[19] H.-J. Böckenhauer, A. Z. M. D. Ullah, L. Kapsokalivas, and K. Steinhöfel, *A Local Move Set for Protein Folding in Triangular Lattice Models*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, vol. 5251, ch. Algorithms in Bioinformatics, pp. 369–381.