Clustered Memetic Algorithm With Local Heuristics for Ab Initio Protein Structure Prediction

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

***Abstract*—Low-resolution protein models are often used within a hierarchical framework for structure prediction. However, even with these simplified but realistic protein models, the search for the optimal solution remains NP complete. The complexity is further compounded by the multimodal nature of the search space. In this paper, we propose a systematic design of an evolutionary search technique, namely the memetic algorithm (MA), to effectively search the vast search space by exploiting the domain-specific knowledge and taking cognizance of the multimodal nature of the search space. The proposed MA achieves this by incorporating various novel features: 1) a modified fitness function includes two additional terms to account for the hydrophobic and polar nature of the residues; 2) a systematic (rather than random) generation of population automatically prevents an occurrence of invalid conformations; 3) a generalized nonisomorphic encoding scheme implicitly eliminates generation of twins (similar conformations) in the population; 4) the identification of a meme (protein substructures) during optimization from different basins of attraction—a process that is equivalent to implicit applications of threading principles; 5) a clustering of the population corresponds to basins of attraction that allows evolution to overcome the complexity of multimodal search space, thereby avoiding search getting trapped in a local optimum; and 6) a 2-stage framework gathers domain knowledge (i.e., substructures or memes) from different basins of attraction for a combined execution in the second stage. Experiments conducted with different lattice models using known benchmark protein sequences and comparisons carried out with recently reported approaches in this journal show that the proposed algorithm has robustness, speed, accuracy, and superior performance. The approach is generic and can easily be extended for applications to other classes of problems.**

***Index Terms*—Clustered memetic algorithm, meme incorporation, pull move, reverse pull move, self-avoiding walk (SAW).**

I. Introduction

PREDICTION of a protein structure is the key to many applications, such as understanding protein function, engineering new proteins, designing drugs, and environmental engineering. Each protein polymer, also known as a polypeptide, consists of a sequence formed from 20 possible amino acids, also referred to as residues. To be able to perform their biological function, proteins fold into a specific minimum energy spatial conformation that becomes the basis of its various functions. In spite of significant ongoing research, the protein structure prediction (PSP) problem is still considered as a grand challenge problem of the century [1]. Further background information on the proteins and the folding process is easily available [2].

In spite of applying simplified models, the search problem remains NP complete [3], i.e., there is no polynomial time solution possible. As a result, numerous nondeterministic optimization algorithms have been investigated, including, e.g., approximation algorithms, Monte Carlo (MC) algorithms, and various evolutionary algorithms (EAs). Approximation algorithms [4], [5], in their attempt to solve the problem in polynomial time, have so far reached only up to 40% of optimal value. Further, due to a rugged landscape, even the traditional MC methods exhibit a tendency to be trapped in the local minima. None of the variants of MC algorithms [6], including, e.g., chain growth algorithms [e.g., prune-enriched Rosenbluth method (PERM)], Markov chains having different temperatures, hybridizing MC, or generalizing ensembles, have been able to handle the multimodal minimization problem. A recent MC algorithm [7] applied a replica exchange with random temperature for improvement, but eventually caused random search movement between the basins of attraction.

Among EAs mimicking nature, the immune algorithm (IA) [8]–[10] with different mutation operators showed limited performance when applied to long sequences. An early genetic algorithm (GA) with MC [11], using internal coordinates, considered only valid conformations having self-avoiding paths as individuals. The standard GA [12], incorporating a preference order encoding for initial individual generation, new fitness function, and allowing nonself-avoiding path (SAW) individuals in population, performed better than GA with MC [11]. A random energy model with a penalty term included in the HP energy function [13] also showed improvement over MC and GAMC [11]. An exhaustive review of evolutionary searches for PSP [14] highlighted the necessity for: 1) improved energy potential; 2) constraint management; 3) correct encoding of conformations; and 4) hybridization. The review on GA [15] also observed that macromutation improved performance, apparently due to random jumping from one basin of attraction to another, but it failed to show better convergence as it did not exploit one basin of attraction completely before jumping to the other and that a one-point crossover is not able to

Manuscript received January 1, 2011; revised August 14, 2011, February 24, 2012, and June 15, 2012; accepted July 9, 2012. Date of publication August 15, 2012; date of current version July 25, 2013.

M. K. Islam is with the Gippsland School of Information Technology, Monash University, Churchill, Vic 3842, Australia, and also with Leap eLearning Pty. Ltd., South Melbourne, Vic 3205, Australia (e-mail: kamrul.islam@infotech.monash.edu.au).

M. Chetty is with the Gippsland School of Information Technology, Monash University, Churchill, Vic 3842, Australia (e-mail: madhu.chetty@infotech.monash.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2012.2213258

transfer the building blocks. Attempts to hybridize GA [16], [17] exhibited slower convergence due to the lack of domain knowledge. Ant colony optimization (ACO) and its variants, e.g., multiple colony distribution [18], [19], flexible ant colony algorithm [20], and hybrid population-based ACO algorithm [21], captured domain knowledge, but failed to handle multiple peaks. Estimation distribution algorithm (EDA) for the protein structure prediction problem [22], [23] replaced the crossover and mutation operation of GA with a probabilistic approach and was successful in capturing the domain information, but failed to systematically switch between multiple basins of attractions.

With these search techniques, protein structure prediction is still limited due to its inherent computational complexity and an astronomically large number of possible solutions. Hence, an in-silico prediction of numerous protein structures still remains a challenge. The complexity and challenging nature of the PSP problem is also highlighted by the Levinthal Paradox [24], which states that, while the nature folds protein sequences spontaneously within microseconds, even an extremely fast computer would take an unbelievably long time to correctly determine a native structure of a small sequence. By taking local optimization decisions first and avoiding a search of irrelevant structures, the structure folds quickly to its native state [25]. Hence, for designing a fast convergence algorithm, it appears crucial to have a combined application of global and local search utilizing domain knowledge.

MA with a powerful combination of global and local searches has been successfully applied to solve complex problems in various domains [26]–[28]. However, its applications to PSP investigations have been surprisingly limited [29]–[32]. Similar to a protein folding phenomena taking place in nature, the memetic algorithm also applies local search first to refine individuals followed by global optimization. In this paper, to emulate nature, we propose a new memetic algorithm for the PSP problem. The novelties of the proposed memetic algorithm are: 1) fitness function incorporating two new terms (H-compliance and P-compliance); 2) dynamic individual generation (DIG); 3) meme generation approach; 4) local optimization with a set of robust pull moves; and 5) population clustering accounting for various basins of attraction in the multimodal search space. The overall algorithm incorporating these novel features is hereby referred to as a clustered memetic algorithm (CMA).

The remainder of this paper is organized as follows. In Section II, we present the background and preliminary studies. This is followed by mathematical explanations underlying the proposed techniques in Section III. In Section IV, different components of the proposed methodology of CMA are explained. The datasets and the setup for simulation experiments along with the results are discussed in Section V. Finally in Section VI, we present conclusions and scope for future work.

II. Background

As stated earlier, a protein polymer consists of a sequence formed from the possible 20 amino acids. To reduce complexity, one restriction that is often made is to restrict the amino acids to a lattice (2-D square, 3-D cube, FCC or face-centered-cubic, and others). These lattice models (simplified protein models), often used for low-resolution protein structure prediction [33], represent amino acids as beads at fixed positions. This reduces the degrees of freedom available to the residues and also the computational complexity, but it still provides structures that can be directly correlated to experimental structures. A further simplification, often implemented, is to categorize the amino acids as either H (hydrophobic) or P (hydrophilic). Restricting the protein conformation to a lattice with amino acids represented as H or P gives the HP model [34]. These simple models [35], [36] allow the development, testing, and comparison of various search algorithms, including those that are studied in this paper. With these models, we can [37], [38] identify quickly and reliably a small set of potential backbone conformations from an otherwise astronomically large and convoluted search space. In this section, we discuss the HP lattice model, including energy functions and encoding techniques.

A. HP and Functional Models

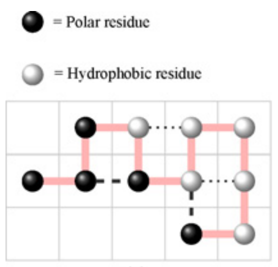
In an HP model, amino acids are classified into either hydrophobic (H) or polar (P), and a protein as a sequence, $s \in \{H, P\}^+$, whose structure has a self-avoiding walk (SAW) requirement [see Fig. 1(a)]. As the residues are located in a lattice, the torsion angles of the peptide bonds between residues are constrained to a finite set of angles determined by the shape of the lattice.

If two residues in the sequence are adjacent to each other (not directly connected) in a lattice, then they are considered topological neighbors (TN). Each occurrence of a TN due to two hydrophobic residues lowers the total energy of the conformation by one [see Fig. 1(b)] in the HP model, whereas it lowers the energy by two in the functional model [see Fig. 1(c)]. In the functional model, a penalty of plus one is applied for all other TN contacts. The PSP problem requires finding a conformation that minimizes its total energy [23]. The energy function of the HP model is traditionally formulated as (1) [see Fig. 1(b)], whereas for the functional model it is given by (2) [see Fig. 1(c)]

$$E_{HP} = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \xi_{jk} \Big|_{\xi_{jk}=\begin{cases} -1, & \text{if } j \text{ and } k \text{ are both H residues and TN} \\ 0, & \text{otherwise} \end{cases}} \tag{1}$$

$$E_{FM} = \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} \xi_{jk} \Big|_{\xi_{jk}=\begin{cases} -2, & \text{if } j \text{ and } k \text{ are both H residues and TN} \\ 1, & \text{otherwise.} \end{cases}} \tag{2}$$

In spite of the simplicity of HP lattice models, the search continues to be NP complete [3]. Further, with the current energy functions not meeting many requirements [39], the search often results in large plateaus [14] and excessive computation time. Therefore, a simple but effective energy function to account for critical effects, e.g., hydrophobic packing [40], [41], is necessary. In this paper, to account for hydrophobic packing, we propose two new energy terms as additional fitness terms for HP models: 1) H-compliance factor



(a)

	H	P
H	−1	0
P	0	0

(b)

	H	P
H	−2	1
P	1	1

(c)

Fig. 1. (a) One possible conformation in HP or functional mode protein for the sequence PPPHPHHHHHP. Topological contact for HH is shown in the dotted lines, whereas other topological contacts are shown in the dashed lines. Total energy for the conformation is −2 in the HP model because there are two HH contacts. Total energy in the functional model is also −2 as the two dotted lines give −4 and the two dashed lines give 2 with a total $-4+2=-2$. Table showing contact energy for the (b) HP model and (c) functional model.

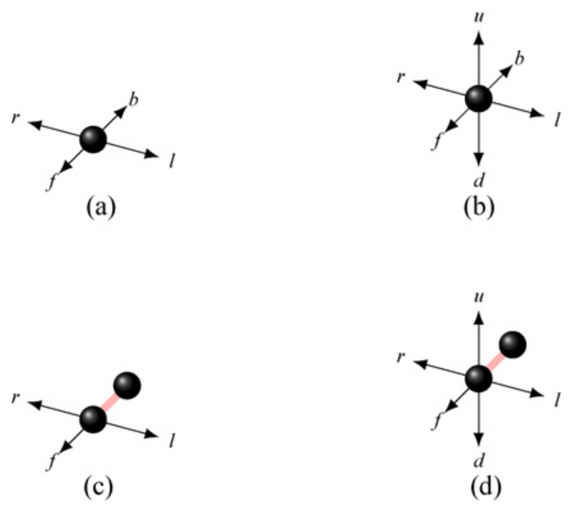


Fig. 2. Absolute and relative encodings. (a) Absolute 2-D square lattice encoding. (b) Absolute 3-D cubic lattice encoding. (c) Relative 2-D square encoding. (d) Relative 3-D square encoding. In absolute encoding, the 2-D square has four possible moves, while the 3-D cubic has six possible moves. In relative encoding the 2-D square has three possible moves, while the 3-D cubic has five.

and 2) P-compliance factor. These terms capture the H and P characteristics of the residues and result in a new fitness function, referred to here as "modified fitness function" (see Section IV-A).

B. Encoding Protein Conformation

Two widely used encodings to construct conformations for protein in the lattice model are: 1) absolute encoding and 2) relative encoding [12], [14]. Along with these, recently proposed nonisomorphic encoding (NIE) [50] has been proven to be a better approach to identify similar conformation. Next, we summarize these approaches.

1) *Absolute Encoding:* Absolute encoding is a direct reflection of the coordinate system where each direction of the coordinate system is represented by a character [11]. As shown in Fig. 2(a) and (b), while the 2-D square lattice requires four characters [f (forward), l (left), r (right), b (back)] for encoding, the 3-D cubic lattice representation requires six characters [f (forward), l (left), r (right), b (back), u (up), d (down)].

2) *Relative Encoding:* Relative encoding reflects the direction of travel on the lattice. From any specific position, a decision for the next position (not considering a back move to ensure self-avoiding walk) is taken [12]. There are three possible directions [F (forward), L (left), R (right)] for a 2-D square lattice, while there are five directions [F (forward), L (left), R (right), U (up), D (down)] for a 3-D cubic, as shown in Fig. 2(c) and (d), respectively.

3) *Nonisomorphic Encoding:* The nonisomorphic encoding dynamically assigns a direction (i.e., up, down, left, or right) relative to the previous moves [42]. This results in a unique correspondence between the encoding and the conformation. Since the existing nonisomorphic algorithm [42] is limited to representing conformation in either a 2-D square or a 3-D cubic lattice, in this paper we have developed a generalized nonisomorphic encoding (GNIE) algorithm applicable to any lattice model (see Algorithm 1).

Consider a conformation of a protein sequence shown in Fig. 3. With absolute encoding, we have eight different representations (chromosomes) corresponding to each of the eight orientations for the same structure, while with relative encoding, there are only two different representations. Thus, relative encoding is usually considered superior to absolute encoding. However, as we can see from Fig. 3, with our GNIE approach (see Section IV-C), all eight orientations have only one encoding (chromosome), i.e., ACBCBDD. Thus, nonisomorphic encoding is able to clearly discriminate between identical conformations and is very effective in reducing multiple occurrences of the same conformation.

C. MA

MAs have been successfully applied to a wide range of optimization problems, e.g., combinatorial, continuous, dynamic, multiobjective [43]. MAs have a flexible architecture that combines the stochastic global search techniques within the framework of EAs and problem-specific local search heuristics [43]. The local search in MA is usually made problem-specific to capture domain-specific knowledge in various ways, such as heuristics, approximation algorithms, different local search techniques, or specialized recombination operator [44].

For the PSP problem under consideration, when a conformation is near-convergence within a basin of attraction, some of the substructures will not change and remain fixed. As explained in Section III, these fixed substructures implicitly contain knowledge about the basin of attraction (i.e., domain-specific knowledge) and are termed memes. They are similar

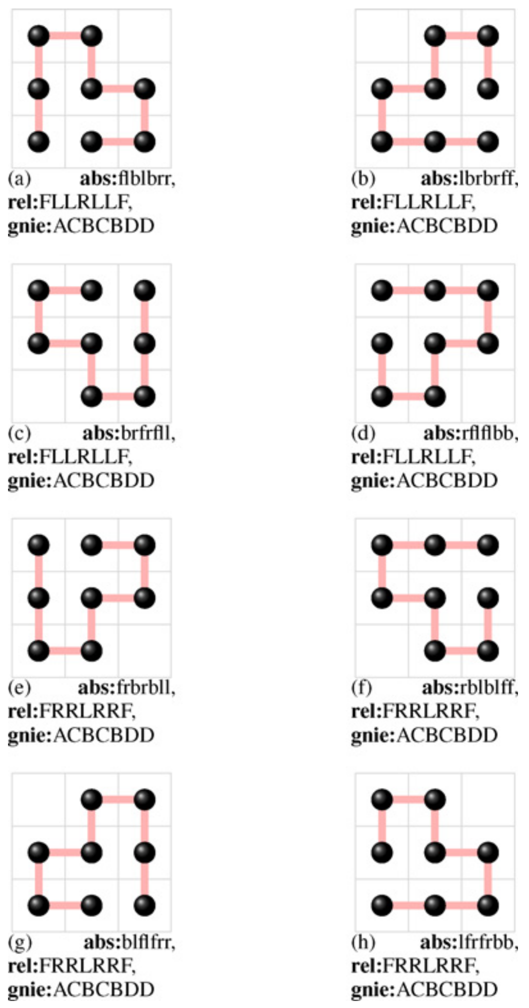

(a) abs:flblbrr, rel:FLLRLLF, gnie:ACBCBDD
(b) abs:lbrbrff, rel:FLLRLLF, gnie:ACBCBDD
(c) abs:brfrfll, rel:FLLRLLF, gnie:ACBCBDD
(d) abs:rflflbb, rel:FLLRLLF, gnie:ACBCBDD
(e) abs:frbrbll, rel:FRRLRRF, gnie:ACBCBDD
(f) abs:rblblff, rel:FRRLRRF, gnie:ACBCBDD
(g) abs:blflfrr, rel:FRRLRRF, gnie:ACBCBDD
(h) abs:lfrfrbb, rel:FRRLRRF, gnie:ACBCBDD

Fig. 3. (a)–(d) Conformations obtained by counterclockwise rotations for an arbitrary sequence. (e)–(h) Respective reflections of conformations in (a)–(d). as: absolute encodings. rel: relative encodings. gnie: nonisomorphic encodings.

to the memes described earlier [45] used for transferring knowledge between individuals. These fixed substructures or memes are similar to the widely known motifs. To understand the concept of a meme, we consider any two randomly chosen protein structures, say *flrrrl* and *frlrrl*, based on an alphabet set $\{f, b, l, r\}$. Here, *rrl* from the fourth to the sixth position can be treated as a meme common to the two conformations. Recalling Holland's definition of schema [46], we note occurrence of $r * l$ to be a schema represented by the alphabet set $\{f, b, l, r, *\}$, where $*$ is a don't-care state normally applied to cover the unrestricted locus of the schema. We can observe here that any meme (such as *rrl*) is essentially a special case of schema (such as $r * l$). Generalized mathematical theory developed for schema in the schemata theorem is, therefore, applicable to memes of the proposed algorithm.

III. Mathematical Preliminaries

As shown later, the technique for generating individuals in the proposed CMA depends on the identification of memes and their applications for generating new individuals. In this section, we mathematically show that although the conformations containing memes undergo normal mutation and crossover operation, the memes survive these operations and continue to contribute toward fast convergence. We also mathematically prove that the memes improve the convergence during local search.

A. Global Search

To understand the survival of a meme during global search G, we take recourse to the well-known Holland's schemata theorem [47] that is based on the concept of schema. However, instead of schema, our analysis here is based on meme (the special case of schema) because the meme concept is central to the current work. In the analysis, we will consider all operations, namely, selection, crossover, mutation, and elitism influencing global search, starting first with the selection process.

1) *Schemata Theorem and Selection:* As mentioned earlier, the memes are formed when the population is close to a convergence region within a basin of attraction. A meme, $\overset{*}{H}_i$, can be defined as

$$\overset{*}{H}_i = < c_1 c_2 \dots c_{o(\overset{*}{H}_i)} >$$

$$\text{where } \begin{cases} o(\overset{*}{H}_i) = \text{length of meme } \overset{*}{H}_i \\ c_i \in \mathbb{C} \text{ and } \mathbb{C} = \text{number of possible moves.} \end{cases} \tag{3}$$

Let $m(\overset{*}{H}_i, t)$ be the number of individuals with meme $\overset{*}{H}_i$ at generation t and $\bar{F}(X^t, t)$ be the observed average fitness of the individuals at generation t. We can calculate $\bar{F}(X^t, t)$ of the population, $P(t) = x_1^t \dots x_{\mathbb{P}}^t$, at generation t as

$$\bar{F}(X^t, t) = \frac{\sum_{j=1}^{\mathbb{P}} F(x_j^t, t)}{\mathbb{P}}$$

$$\text{where } \begin{cases} x_j^t \in P(t) \\ \mathbb{P} = \text{ number of individuals in the population.} \end{cases} \tag{4}$$

Similarly, the average fitness of individuals with meme $\overset{*}{H}_i$ at time t, $\bar{F}(\overset{*}{H}_i, t)$, can be represented as

$$\bar{F}(\overset{*}{H}_i, t) = \frac{\sum_{x_j^t \in \mathbf{H}} F(x_j^t, t)}{m(\overset{*}{H}_i, t)}$$

$$\text{where } \begin{cases} \mathbf{H} = \text{ set of all individuals with the meme in the population} \\ m(\overset{*}{H}_i, t) = \text{ number of individuals with meme } \overset{*}{H}_i. \end{cases} \tag{5}$$

To calculate $m(\overset{*}{H}_i, t{+}1)$, the expected number of individuals with meme $\overset{*}{H}_i$ at generation $t + 1$, we ignore the effect of crossover and mutation. As a result, we get the expected value as

$$\begin{aligned} m(\overset{*}{H}_i, t + 1) &= \mathbb{P} \frac{\sum_{x_j^t \in \mathbf{H}} F(x_j^t, t)}{\sum_{j=1}^{\mathbb{P}} F(x_j^t, t)} \\ &= \frac{\sum_{x_j^t \in \mathbf{H}} F(x_j^t, t)}{\bar{F}(X^t, t)} \\ &= \frac{\bar{F}(\overset{*}{H}_i, t) m(\overset{*}{H}_i, t)}{\bar{F}(X^t, t)}. \end{aligned} \tag{6}$$

The expected number of individuals given in (6) will be used later in (14) to determine the expected existence of memes in subsequent generations. Before that, let us consider the effect of crossover and mutation that can lower the number of instances of memes $\overset{*}{H}_i$ in the population.

2) *Crossover:* Crossover operation will disrupt memes if the crossover point for two selected individuals lies within the memes where the length of a meme is given as $o(\overset{*}{H}_i)$. So, the probability that a meme will be disrupted by a single-point crossover is $\mathfrak{D}_p = \frac{o(\overset{*}{H}_i)}{n-1}$. If the crossover probability is P_c, then the lower bound of the probability of existence of meme after crossover will be

$$\mathfrak{E}_p \geq (1 - P_c\mathfrak{D}_p) = \left(1 - P_c\frac{o(\overset{*}{H}_i)}{n-1}\right). \tag{7}$$

This is the lower bound of existence of a meme because there are three scenarios in which memes are not disrupted even though the crossover point lies between the memes.

a) *Parents with the same memes:* With both parents having the same meme, the crossover point, even if it lies between the memes, would cause no disruption provided the SAW constraint is not violated. If A is the probability of such parents, then for a given parent with meme $\overset{*}{H}_i$, the probability that the second parent will have the same meme is

$$A = \frac{1}{|\mathbb{C}|^{o(\overset{*}{H}_i)}}\Big|_{\text{where } \mathbb{C}=\text{number of possible moves.}} \tag{8}$$

b) *Parents with partial memes:* If meme $\overset{*}{H}_i$ is viewed as concatenation of two partial memes $(H_1,\ H_2)$, then generation of $\overset{*}{H}_i$ is possible provided the partial memes are contained in the two selected parents. The possible ways B to form a meme $(\overset{*}{H}_i)$ from two parts $(H_1,\ H_2)$ of the parents are obtained as follows:

$$\begin{aligned} B &= \sum_j^{o(\overset{*}{H}_i)} \frac{1}{\mathbb{C}^j}\frac{1}{\mathbb{C}^{o(\overset{*}{H}_i)-j}}_{\text{where } \mathbb{C}=\text{number of possible moves}} \\ &= \sum_j^{o(\overset{*}{H}_i)} \frac{1}{\mathbb{C}^{o(\overset{*}{H}_i)}} \\ &= \frac{o(\overset{*}{H}_i)}{\mathbb{C}^{o(\overset{*}{H}_i)}}. \end{aligned} \tag{9}$$

c) *SAW constraint:* Any individual generated by crossover is valid provided it meets the SAW requirement. Due to this constraint, generation of a number of individuals gets restricted. If the probability of generating an individual with SAW is P_{saw}, then the individuals generated by disruption of memes may not be SAW. The probability, C, that the disrupted meme is not a SAW is as follows:

$$C = (1 - P_{saw})\mathfrak{D}_p. \tag{10}$$

Considering these scenarios, the meme disruption probability of (7) will be reduced to a crossover operation as follows:

$$\begin{aligned} \mathfrak{E}_{p_t} &= 1 - P_c(\mathfrak{D}_{p_t} - \psi_t)\big|_{\text{where } \psi_t=A+B+C}. \\ &= 1 - P_c\left(\frac{o(\overset{*}{H}_i)}{n-1} - \frac{1}{|\mathbb{C}|^{o(\overset{*}{H}_i)}} - \frac{o(\overset{*}{H}_i)}{\mathbb{C}^{o(\overset{*}{H}_i)}} - (1 - P_{saw})\mathfrak{D}_p\right). \end{aligned} \tag{11}$$

Equation (11) shows that a shorter length meme will have a better chance of surviving even if the crossover point lies between memes.

3) *Mutation:* Mutation is performed by flipping a bit within a meme that can also cause its disruption. If μ_t is the probability of mutation at generation t, then the probability that a bit is not changed is given as $(1 - \mu_t)$. The probability of survival of meme of order $o(\overset{*}{H}_i)$ is given as

$$\begin{aligned} \mathfrak{E}_{\mu_t} &= (1 - \mu_t)^{o(\overset{*}{H}_i)} \\ &= (1 - o(\overset{*}{H}_i)\mu_t)\big|_{\text{expanding and ignoring higher powers of } \mu_t}. \end{aligned} \tag{12}$$

Clearly, a fixed rate of mutation will result in disruption of memes and can randomly move individuals to other unknown regions. To prevent this and increase the chance of survival of memes subjected to mutation, we propose here an adaptive mutation approach. The adaptation of mutation rate μ_t at any iteration (time) t is carried out by starting with a higher mutation rate initially and systematically decreasing its value as follows.

The adaptive mutation rate [see (13)] at the tth iteration, μ_t, is reduced gradually applying two reduction terms: one is proportional to mutation rate $\mu_{(t-1)}$ at the $(t-1)$th iteration, multiplied by $\frac{\delta_t}{F_t^*}$, and the other is a term inversely proportional to the iteration count, t. The first term is applied only when $\mu_{(t-1)}$ is greater than 0 and the second term is applied only if $t > \varrho$. We keep $\varrho = \frac{1}{b\times\mu_0}$, where b is a constant greater than 1. Here, δ_t is defined as a fitness difference between the best individual at any iteration (t) and the $(\mathbb{P}\times\wp)$th best individual of that iteration.

From (13) (see top of the next page), we note that the mutation rate will reduce to zero as the difference in fitness, δ_t, of the individuals in a population at any iteration t tends to zero. This will happen when the population enters into a basin of attraction and, in turn, converges to any local minima. This ensures that $\mathfrak{E}_{\mu_t} \approx 1$ of (12), which actually guarantees that the impact of mutation on the disruption of meme is minimum.

4) *Elitism:* Elitism that involves passing a small number of highly fit individuals into subsequent generations can also influence the survival of memes. If the number of elite individuals is $\mathcal{E}$ and the percentage of elite individuals containing the meme is χ (see Section IV-D1), then the number of elite individuals with the meme is $(\mathcal{E}\times\chi)$.

To find out the expected existence of memes in a subsequent generation in conformations, we again consider (6) and note that

$$\begin{aligned} M(H, t+1) &= \forall_i m(\overset{*}{H}_i, t+1) \\ &\geq \prod_i \frac{\bar{F}(\overset{*}{H}_i, t)}{\bar{F}(X^t, t)} m(\overset{*}{H}_i, t)\mathfrak{E}_{p_t}\mathfrak{E}_{\mu_t} + \mathcal{E}\times\chi. \end{aligned} \tag{14}$$

From (14), we observe that, as $\bar{F}(\overset{*}{H}_i, t) > \bar{F}(X^t, t)$, there will be exponentially more memes with a lower length [48]. This is because the number of individuals that will have all the memes from the basins of attraction will increase because $\mathfrak{E}_{p_t}$ and $\mathfrak{E}_{\mu_t}$ have values close to 1 when the search is near convergence.

$$\mu_t = \mu_{(t-1)} - \frac{\delta_t}{F_t^*}\mu_{(t-1)}\Big|_{\left\{\substack{\text{if } \mu_{(t-1)}>0 \\ \text{and } \delta_t \neq \delta_{t-1}}\right.,\ t=m,m+1,\ldots} - \frac{1}{t}\Big|_{\left\{\substack{\text{if } t>\varrho \\ \varrho=\text{Time to Convergence}}\right.},$$
$$\delta_t = F_t^* - F(\mathbb{P} \times \wp)\Big|_{\text{where } \left\{\substack{\mathbb{P}=\text{Number of individuals} \\ \wp=\text{Required fraction in the BOA}}\right.}. \tag{13}$$

Thus, we see that memes will survive even with the crossover, mutation, or applications of elitism.

B. Local Search

When the search is near convergence to a local minima, it actually enters within a specific basin of attraction [49]. Goldberg and Voessner [49] showed that a search space can be divided into several basins of attractions. To examine how the memes contribute to speeding up the convergence process, let n_{C_β} be the number of conformations within a basin of attraction β. Let ε be the set of memes in the basin of attraction β. If $l(\varepsilon_i)$ gives the length of the ith meme, then the total length of all the memes of that basin of attraction will be $\sum_{1\le i\le|\varepsilon|} l(\varepsilon_i)$. Now, to find out how many conformations actually contain memes, let us define the following terms:

$\mathbb{C}$ = set of all possible moves;
$\mathbb{L}_c$ = length of the chromosome of a conformation;
$|\overset{*}{\mathbb{C}}|$ = number of conformations with memes.

If Ω defines all possible conformations in a search space, then

$$|\Omega| = |\mathbb{C}|^{\mathbb{L}_c}. \tag{15}$$

Similarly, the number of conformations with memes falling within the same basin of attraction β is

$$|\overset{*}{\mathbb{C}}| = |\mathbb{C}|^{\mathbb{L}_c - \sum_{1\le i\le|\mathbb{C}|} l(\varepsilon_i)}. \tag{16}$$

From (15) and (16), it is clear that $|\overset{*}{\mathbb{C}}| \le |\Omega|$. The probability that an individual will fall in that basin of attraction is

$$P_\beta = \frac{|\beta|}{|\Omega|}\Big|_{\text{where } \left\{\substack{|\beta|=\text{ number of individuals in the basin} \\ |\Omega|=\text{ number of individuals in the search space}}\right.} \tag{17}$$

whereas the probability of an individual with a meme to fall under the basin of attraction β is

$$\overset{*}{P}_\beta = \frac{|\beta|}{|\overset{*}{\mathbb{C}}|}. \tag{18}$$

From (17) and (18), we can easily infer that $\overset{*}{P}_\beta \ge P_\beta$ as $|\overset{*}{\mathbb{C}}| \le |\Omega|$. Thus, we can clearly see that applying meme concepts will result in a faster convergence within a basin of attraction.

IV. Method

Our proposed novel MA approach uses an improved fitness function based on two factors: H-compliance and P-compliance. Furthermore, it develops clusters of memetic algorithms, and identifies and transfers memes while seeding new individuals in the main cluster of CMA. The entire methodology is explained as follows.

A. Energy Function

To capture the H and P characteristics of the residues and account for hydrophobic packing, we include two new energy terms: 1) H-compliance factor and 2) P-compliance factor. These terms result in a new fitness function that we will refer to, henceforth, as modified fitness function (MFF).

1) *H-Compliance:* We know that all H residues tend to position themselves within the protein core and stay close to the H-core center. As a measure of how well this requirement is complied, we define a new term called H-compliance, which is measured as a radial distance of H residue from H-core center. Thus, residues closer to the H-core center will be rewarded with smaller H-compliance values. Similarly, summing the H-compliance of individual H residues in a sequence will define the H-compliance of an entire conformation under consideration.

As shown in Fig. 4(b), to calculate H-compliance, we determine the coordinate (x_c, y_c, z_c) of the center of an imaginary cube forming the H core as $x_c = (x_{h_{\max}} - x_{h_{\min}})/2$, $y_c = (y_{h_{\max}} - y_{h_{\min}})/2$, and $z_c = (z_{h_{\max}} - z_{h_{\min}})/2$. Further, for any ith H-type residue, if the H-compliance is h_i and its coordinate is (x_{hi}, y_{hi}, z_{hi}), the overall H-compliance of the jth conformation will be $H_j = \sum_{i=1}^{n_h} h_i$, where $1 \le j \le \mathbb{P}$ and $\mathbb{P}$ indicates population size. That is

$$H_j = \sum_{i=1}^{n_h} h_i = \sum_{i=1}^{n_h} (x_c - x_{hi})^2 + (y_c - y_{hi})^2 + (z_c - z_{hi})^2. \tag{19}$$

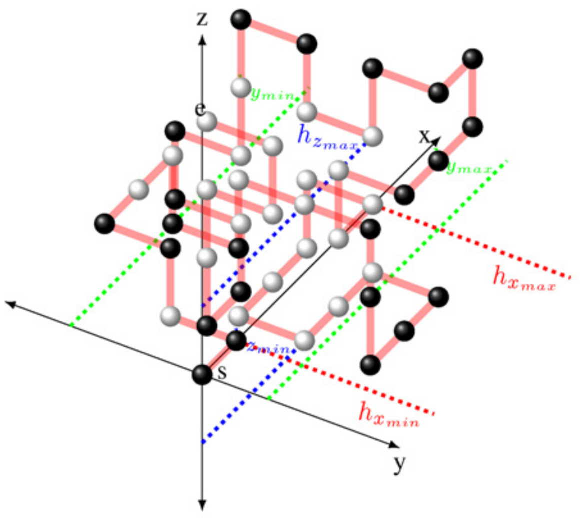
From the H-compliance given by (19), we compute the following $E_j^{H\text{-compliance}}$ as the average of the H-compliance of the conformation

$$E_j^{H\text{-compliance}} = H_j/n_h. \tag{20}$$

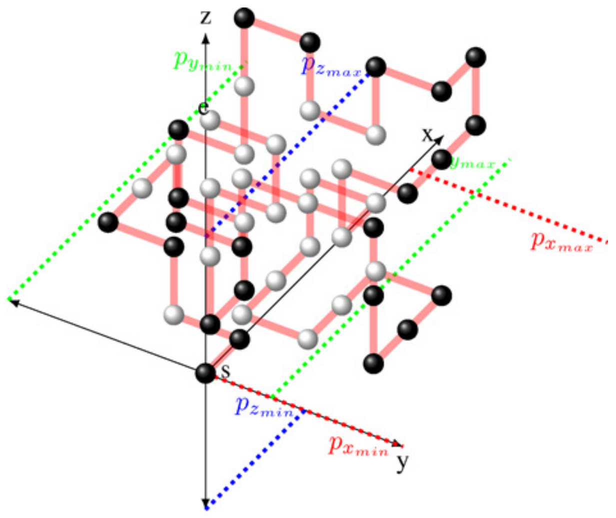
Here, n_h is the total number of H-type residues in the sequence. $E_j^{H\text{-compliance}}$ is defined above and is added as an additional fitness term to the function of (1).

2) *P-Compliance:* P residues tend to be as much away from the H-core center as possible. To comply with this requirement, we define a term P-compliance as a measure of how close the P residue is to any of the sides $x_{p_{\max}}$, $y_{p_{\max}}$, $z_{p_{\max}}$, $x_{p_{\min}}$, $y_{p_{\min}}$, $z_{p_{\min}}$ of a P-boundary cube, as shown in Fig. 4(b). We have chosen a P-boundary cube rather than a H-core here because the P residues are located close to the outer periphery and are thus not measurable from the H-core center. The smaller the value of P-compliance, the closer it is to the P-boundary cube. The sum of the P-compliances of all the P-type residues gives the P-compliance of the conformation under consideration.

As shown in Fig. 4(b), for measuring the P-compliance p_i, we determine the minimum distance of an ith P-type residue from the P-boundary cube. With coordinates of ith P residue given as (x_{pi}, y_{pi}, z_{pi}), the P-compliance of the jth



(a) H-Compliance



(b) P-Compliance

Fig. 4. Imaginary boundary cube for (a) H-residues and (b) P-residues. The conformation is taken for sequence $B5$ (see Table II) on a 3-D cubic lattice. Here, ○ stands for hydrophobic residue and ● stands for polar residue. Again, s = start point and e = end point.

conformation, where $1 \leq j \leq \mathbb{P}$ and $\mathbb{P}$ indicates population size, is given as

$$P_j = \sum_{i=1}^{n_p} p_i = \sum_{i=1}^{n_p} \big(\min\{|x_{p_{\min}} - x_{pi}| \\ |x_{p_{\max}} - x_{pi}|, |y_{p_{\min}} - y_{pi}|, \\ |y_{p_{\max}} - y_{pi}|, |z_{p_{\min}} - z_{pi}|, |z_{p_{\max}} - z_{pi}|\}\big). \quad (21)$$

Again, the corresponding fitness term $E_j^{P\text{-compliance}}$ is defined as the average P-compliance of the conformation. The term n_p is the total number of P residues in the individual

$$E_j^{P\text{-compliance}} = P_j / n_p. \quad (22)$$

Thus, the complete MFF for the jth conformation is given as follows:

$$E_j^{\text{MFF}} = aE_j^{TN} + E_j^{H\text{-compliance}} + E_j^{P\text{-compliance}}. \quad (23)$$

Here, E_j^{TN} is the original fitness for the jth confirmation computed from (1). The value of constant a, appearing as the multiplier of E_j^{TN}, is kept high to ensure that E_j^{TN} maintains an influential effect on the MFF.

This generic fitness function of (23) is applied for all experimental studies reported here.

B. Individual Generation

With any known stochastic search applied for the PSP problem, the individuals in the population are randomly generated. Once generated, the individual is then validated to confirm that it is SAW compliant. If not, it is discarded and a new individual is again generated randomly. Hence, this process makes the generation of individuals inherently computationally intensive, especially for long sequences. To overcome this limitation, which slows down the search algorithm significantly, we propose a DIG approach. Not only does DIG guarantee SAW compliance, it also ensures the process to be significantly faster than traditional approaches (details are given in our previous work [50]).

C. Encoding Conformations

Of the three encoding schemes described in Section II-B, the GNIE results in a unique code for a given conformation. For example, in Fig. 7(a) and (b), a unique GNIE of ACADADB-DBCB for two different orientations of the same conformation is shown. In contrast, relative encoding results in two different codes, namely FRLLRLLRLLR and FLRRLRRLRRL, for the same conformation. However, the existing NIE technique [50] can be applied for only either a 2-D square or 3-D cubic lattice. To generalize the technique and make it suitable for any lattice representation, we propose here a new GNIE. The GNIE utilizes the symmetry (each move has an equivalent move in the opposite direction) that exists in all lattice models. The pseudocode for the GNIE algorithm is shown as Algorithm 1. GNIE starts by inserting the first character as A (line 4) and keeps track, in an array, of all different direction vectors found *directionAndMoves* (see lines 5, 8, 15, and 18). The method *findNonIsoEncode* looks for a vector in the *directionAndMoves* array. If found, the corresponding character is returned. Otherwise, it returns *null* (see line 11).

D. Memes

As the population begins to converge, it is seen that some regions (i.e., substructures) in the individuals tend to remain fixed. As shown mathematically in Section III-B, once they are formulated, these memes reduce the search space and thereby speed up the convergence. A systematic identification and fixing of these memes causes the memes to implicitly contain the vital domain-specific knowledge (i.e., occurrence of specific substructures in specified locations). The process of meme identification involves simply finding which substructure best suits a particular segment of the conformation. In fact, the entire process is analogous to template-based modeling (i.e., threading technique), where a suitable substructure is chosen from a pool of similar structures [51]. If the set of memes are identified from different basins of attraction, these can be treated as a

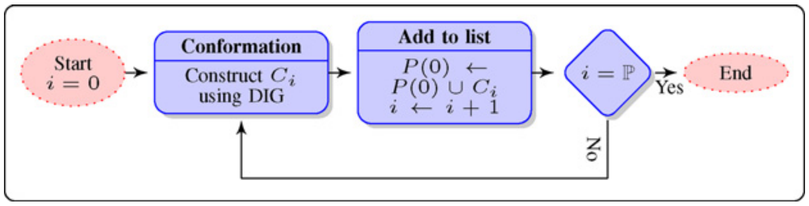


Fig. 5. DIG initialization process in a population-based EA. $P(0)$ = initial population. C_i = ith conformation.

Algorithm 1 Generic nonisomorphic encoding (GNIE)

```
1: procedure GetNonIsoMorphicEncoding points (An array of points for a conformation)
2:   $startWith$ = 'A'                    ▷ First move always A
3:   $\vec{V_1} \leftarrow points[2] - points[1]$
4:   $encoding \leftarrow startWith$
5:   $directionAndMoves.Add(startWith, \vec{V_1})$
6:   $startWith \leftarrow startWith + 1$  ▷ Proceed to next character
7:   $\vec{V_2} \leftarrow points[1] - points[2]$
8:   $directionAndMoves.Add(startWith, \vec{V_2})$
9:   for $i \leftarrow 3, points.length$ do
10:      $\vec{V_1} \leftarrow points[i] - points[i-1]$
11:      if $(c \leftarrow findNonIsoEncode(directionAndMoves, \vec{V_1})) \neq null$ then
12:          $encoding \leftarrow encodings + c$
13:      else
14:          $startWith \leftarrow startWith + 1$   ▷ Proceed to next character
15:          $directionAndMoves.Add(startWith, \vec{V_1})$
16:          $encoding \leftarrow encodings + startWith$
17:          $\vec{V_2} \leftarrow points[i-1] - points[i]$
18:          $directionAndMoves.Add(startWith, \vec{V_2})$
19:      end if
20:  end for
21:  return $encoding$   ▷ Returns nonisomorphic encoding
22: end procedure
```

database or template for transferring knowledge from these basins of attraction to the new individuals that are generated.

It is also important to estimate the condition during evolution when memes can be identified and fixed. For this, we dynamically compute the difference in fitness Δ for all those individuals having close fitness values that do not change for a specified number of iterations, n_c. The criteria of convergence C_{local} is given by following (24) (shown on the next page), applied to the population $P(t) = x_1^t \ldots x_{\mathbb{P}}^t$.

Here, C_{local} identifies the convergence of population P to any local minima based on two main criteria: 1) the best fitness value in the population remains fixed for n_c generations and 2) the difference in fitness value of the top 75% (i.e., $\frac{3}{4}$th of total $\mathbb{P}$ individuals) of the best fitted individuals is within a close range, Δ.

1) *Identification:* For this paper, we propose to identify a meme by finding a highly probable move set having at least two consecutive moves (involving three residues), each of which has an independent probability greater than a specified threshold. The meme structure description will be a 3-tuple containing the move set and its start and fitness. Moreover, while generating memes, it is also essential to identify and mark those individuals as to which specific basin of attraction they belong.

In general, consider a conformation C_i in any lattice model and $\mathbb{C}$ to be a set of all possible moves with size $|\mathbb{C}|$, then $\mathbb{C}_q \in \mathbb{C}$ with $q = 1, 2, \cdots, |\mathbb{C}|$. Although the proposed meme technique is generic and applicable to any lattice model, for the sake of explanation, let us consider a 2-D lattice with a relative encoding. Considering relative 2-D encoding, $|\mathbb{C}| = 3$, and we have the set of possible moves as ($\mathbb{C}_0 = F, \mathbb{C}_1 = L, \mathbb{C}_2 = R$). If $\mathbb{L}$ is the length of the sequence, then conformation will have only ($\mathbb{L} - 2$) moves [52], because the first move is always treated as F (forward).

Next, we construct a 2-D matrix M_{Ci} with rows defining the ($\mathbb{L} - 2$) positions of moves and the columns describing the actual moves $\mathbb{C}_q$. Thus, the size of matrix M_{Ci} will be $(\mathbb{L} - 2) \times |\mathbb{C}|$. The matrix M_{Ci} is populated as $M_{Ci} = [a_{rq}]_{r=1,\ldots,\mathbb{L}-2,q=1,\ldots,|\mathbb{C}|}$. Now, if the rth position of a conformation C_i is $\mathbb{C}_q$, then $a_{rq} = \epsilon \times F(C_i)$; otherwise, $a_{rq} = 0$. The constant $\epsilon = -1$ and $F(C_i)$ is the fitness of the ith conformation.

To determine a highly probable meme that is likely to occur in a subsequent generation, we obtain a matrix $\Gamma = \sum_{i=1}^{\mathbb{P}} M_{Ci}$ with the summation taking place on the entire population, $\mathbb{P}$. Multiplying Γ with a column vector $[1 \quad 1 \quad 1]^T$ results in another column vector $\Lambda = \Gamma[1 \quad 1 \quad 1]^T = [\rho_1 \quad \cdots \quad \rho_{l-2}]^T$. The rth row of Λ represents the cumulative weight, ρ_r of rth position for all conformations. To obtain the probability of occurrence of each move at this rth position, we multiply the rth row of matrix Γ by $(1/\rho_r)$ and obtain another matrix Γ'. The matrix Γ' contains significant information about the probability of occurrence of moves at a given position. To classify a move in a given position as highly probable, we define a cutoff value

$$\chi = 0.5 + \frac{1}{|\mathbb{C}|}. \tag{25}$$

If at least two consecutive moves (three residues) in the matrix Γ' have value greater than χ, then this highly probably set of moves is identified as a meme.

Meme identification can be explained by an illustrative toy sequence HPHPPHHPHP of length 10 (i.e., nine moves). Let us consider any three randomly selected conformations, i.e., FLFLLRRLR, FFLLFFRLR, and FLFLLRRFF, having fitness of −2, −1, and −2, respectively. The resulting (9×3) matrix and the step-by-step changes of the matrix for the three conformations are shown in a tabular form in Table I. The matrix is populated by the fitness function, as explained next. To illustrate, let us randomly choose a position (e.g., position 3) of the conformation. We see that for the first conformation FLFLLRRLR, it has a $\epsilon \times F(C_i) = 2$ and has F move in the third position. Hence, 2 is recorded in

$$C_{\text{local}}(P) = \begin{cases} 1, & \text{if} \quad \begin{cases} F^*(P(t)) = F^*(P(t+n_c)) = F \\ F^*(P(t+n_c)) - F^{\frac{3}{4}}(P(t+n_c)) = \Delta \end{cases} \Bigg|_{\text{where}} \begin{cases} F^* = \text{returns best fitness value in a population} \\ F^{\frac{3}{4}} = \text{returns fitness of } \frac{3}{4}\mathbb{P}\text{th individual} \end{cases} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

position 3 of FLFLLRRLR in the matrix. Similarly, for the second conformation FFLLFFRLR, we record 1 under L move for position 3. Further, 2 recorded under the F column for the first conformation is also carried forward and copied under the F column of conformation 2. Now, for the third conformation, we note that its $\epsilon \times F(C_i) = 2$, and it has an F move in position 3. Since a value of 2 is already recorded under F from the first conformation, we record $4(= 2 + 2)$ in its third position under F. Thus, all the positions for all the conformations are updated. The guideline for the best probable conformation in the next generation is obtained by calculating the percentage of occurrence of the moves for a given position. In the example discussed above, we can see from the table that the total value of position 3 for F, L, and R is $5(= 4+1)$. It has an occurrence of F as 80% (i.e., $4/5 \times 100 = 80$), an occurrence of L as $20\%(1/5 \times 100 = 20)$, and an occurrence of R as 0%.

Now, if we consider the cutoff $\chi = 0.8$, then we will find that FLFLLRR (from the first position to the seventh position) qualifies to be a meme. Generating memes in this manner with highly probable moves, the resulting memes capture the domain-specific knowledge (i.e., the most probable substructures) of a particular basin of attraction where they occur.

2) *Meme-Based Individual Generation:* First, an individual is generated using DIG and then each meme of a particular cluster is threaded in its corresponding position in the individual. If a meme fails to produce a valid individual (i.e., an individual with SAW), then its reflection is tried instead. A successful threading of a meme in the randomly generated individual results in a valid individual in the population. A specific threading process is considered unsuccessful if none of the memes of the cluster is able to produce an individual with SAW. In that case, the individual is discarded and a new individual is generated using DIG, and a similar threading process is carried out with another meme in its corresponding location in the manner discussed above. The meme-based random individual generation process is shown as a flowchart in Fig. 6.

3) *Encoding:* Consider the three identical memes shown in Fig. 7(c) within dashed rectangles. Although the memes are occurring at different locations in the sequence, the memes should have a unique code as they are identical. While with the GNIE approach, these identical memes are coded differently as *CAD*, *ADB*, and *DBC*; with relative encoding, all memes are uniquely encoded as *RLL*. This problem with GNIE occurs because the potential moves in nonisomorphic encodings depend on both the current and previous moves, while with relative encoding they depend only on the current move. Since a meme can only be inserted in any conformation if they are independent of previous moves, we have coded them with a relative encoding technique that then allows us to insert them without any conversion. With any other encoding technique, a translation of a move set would be necessary before insertion. We thus note that while GNIE is suitable for encoding conformations, relative encoding is necessary for encoding memes.

E. Pull Move

Since the introduction of a pull move by Lesh *et al.* [53], many applications have been reported [16], [54], which applied and modified the pull move. Here, we apply a robust pull move technique as a local search. The technique is based on identifying 12 different scenarios for applications of pull move in the forward direction and 12 more scenarios for the reverse pull move (see [50] for details). Further, we also consider two more pull move scenarios that will pull the chain from both ends, namely, begin pull and end pull, which are controlled by the following definitions.

Definition 1: (Conformation)
Let V be the set of vertices of residues and E be the set of lines connecting the residues where the length of each line can be defined as $\forall_{0<j<\mathbb{L}}|(A_j, A_{j+1})| = 1$ and each vertex of residue can be defined as $\forall_{1\leq i\leq\mathbb{L}}\{A_i = (x_i, y_i)\}$.

Thus, a conformation, $C_{\mathcal{L}}$, in any lattice, $\mathcal{L}$, is an ordered pair $C_{\mathcal{L}} = (V, E)$, where $V = \{A_1, A_2, \ldots, A_{\mathbb{L}}\}$ with $\mathbb{L}$ being the number of amino acids and $E = \forall_{1<i<\mathbb{L}}\{(A_{i-1}, A_i) \cup (A_i, A_{i+1})\}$.

Definition 2: (Begin Pull)
A begin pull move is possible if there exists an empty space in the neighborhood of A_1 or A_2 in any lattice model. In begin pull, the A_1 moves to the empty space and the chain is adjusted to allow forward pull move.

Definition 3: (End Pull)
An end pull move is possible if there is an empty space in the neighbor of $A_{\mathbb{L}-1}$ in any lattice model. In end pull, the $A_{\mathbb{L}}$ moves to the empty space and the chain is adjusted to allow reverse pull move.

F. Clustering Memetic Algorithm

The search space for the complex PSP problem is multimodal and has many basins of attraction. Due to the phenomena of genetic drift, stochastic variations caused by the genetic operators can result in a population drift to any of these peaks [55]. In effect, this may result in either slowing down the convergence or, worse still, getting stuck in a local optimum.

Increasing the size of the initial population to cover all possible peaks is not desirable, because it will result in a very large computational time [56], [57]. Clustering of data will help identify different basins of attraction (BOA) effectively and efficiently [58]. To illustrate the process, let us segregate the search space in a manner similar to that followed by Goldberg and Voessner [49]. The search space Ω is divided into $|\beta|$ number of BOA such that $\Omega = \bigcup_i \beta_i$. For the sake of

TABLE I
ILLUSTRATING THE GRADUAL CHANGE IN THE WEIGHTED MATRIX WITH A TOY SEQUENCE HPHPPHHPHP

Pos	FLFLLRRLR (-2)			FFLLFFRLR (-1)			FLFLLRRFF (-2)			%		
	F	L	R	F	L	R	F	L	R	F	L	R
1	2			3			5			1	0	0
2		2		1	2		1	4		0.2	0.8	0
3	2			2	1		4	1		0.8	0.2	0
4		2			3			5		0	1	0
5		2		1	2		1	4		0.2	0.8	0
6			2	1		2	1		4	0.2	0	0.8
7			2			3			5	0	0	1
8		2			3		2	3		0.4	0.6	0
9			2			3	2		3	0.4	0	0.6

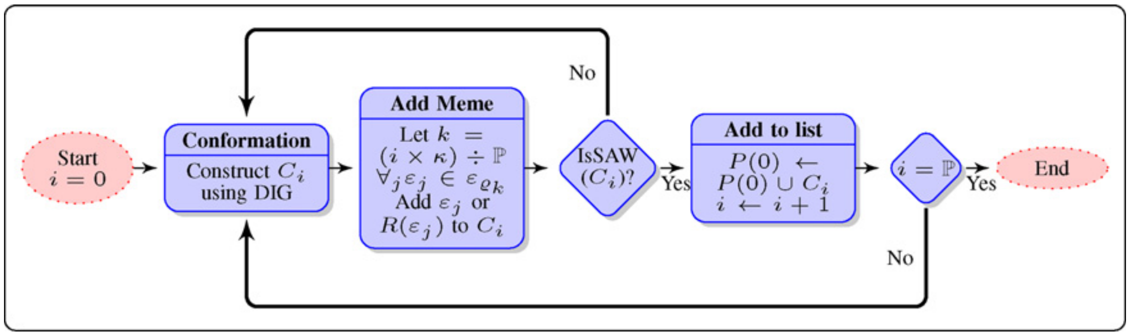

Start
$i = 0$
Conformation
Construct C_i using DIG
Add Meme
Let $k = (i \times \kappa) \div \mathbb{P}$
$\forall_j \varepsilon_j \in \varepsilon_{\varrho_k}$
Add ε_j or $R(\varepsilon_j)$ to C_i
IsSAW (C_i)?
No
Yes
Add to list
$P(0) \leftarrow P(0) \cup C_i$
$i \leftarrow i + 1$
$i = \mathbb{P}$
Yes
No
End

Fig. 6. Meme base random individual generation process. The meme is incorporated into the individual after generating a random individual using DIG. An equal number of individuals is generated for each cluster. ε_j = jth meme. $P(0)$ = population. C_i = ith conformation. ϱ_k = kth cluster.

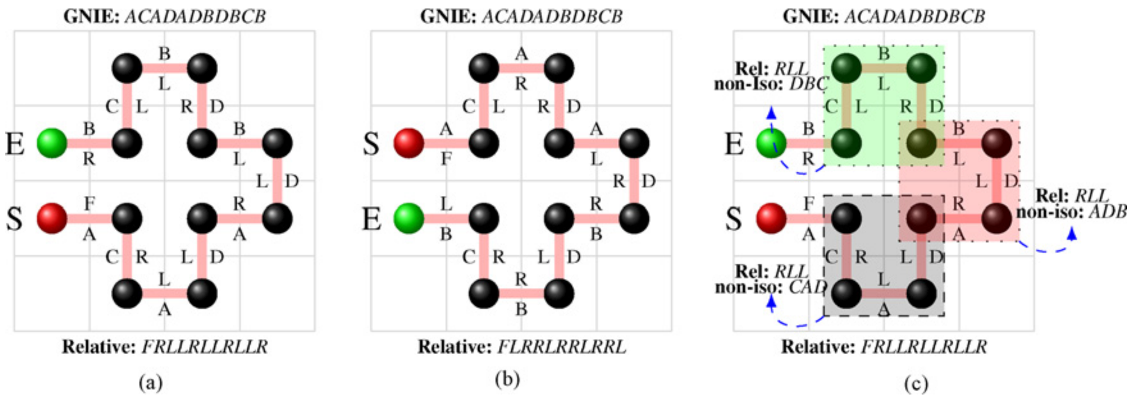


Fig. 7. (a) and (b) Relative and nonisomorphic encoding is given for a conformation. While no change in encoding occurs for nonisomorphic encoding, the relative method incorrectly changes the encoding for the same conformation with different orientations. (c) This comparison of encoding of memes is of relative and nonisomorphic approaches. Here, no change in meme encoding occurs for the relative approach, while the nonisomorphic approach shows different encoding for the same meme but with different orientations. Legend: 🔴 = start point, 🟢 = end point.

simplification, if we assume that all BOA are of equal size, the probability of getting trapped in any of the BOA by any of the clusters is $P_\beta = \frac{1}{|\beta|}$. Hence, by definition, each cluster of CMA will converge to any of the BOA with the probability $\frac{1}{|\beta|}$. This means that, considering a normal distribution, the proposed CMA will increase the probability of selecting different BOA, P_β^{CMA}, as follows:

$$P_\beta^{\text{CMA}} = \begin{cases} \frac{\kappa}{|\beta|}\Big|_{\kappa=\text{total number of clusters},} & \text{if } \kappa < |\beta| \\ 1, & \text{otherwise.} \end{cases} \tag{26}$$

Again, if the BOAs are not of the same size, then the probability, $P(\beta_i)$, that a cluster will converge to a BOA, β_i, will depend on the size of the BOA, which is

$$P(\beta_i) = \frac{|\beta_i|}{|\Omega|} \quad \text{where} \quad \sum_i P(\beta_i) = 1.$$

To effectively search the whole search space, a minimum of

$$\kappa_{\min} = \frac{|\Omega|}{\beta_{\min}}\Big|_{\beta_{\min}=\arg\min\{\forall_i|\beta_i|\}} \tag{27}$$

clusters are required. Number of clusters greater than $\kappa_{\min}$ will not be advantageous in the search process.

The process of meme identification is analogous to finding which substructure best suits the particular segment of the conformation [51]. In our approach, the collection of memes identified from different clusters serves as a database of

TABLE II
WIDELY USED BENCHMARK SEQUENCES FOR THE HP PROTEIN MODEL

Seq	Len.	Sequence	$E^*_{(2\text{-}D\,sqr)}$	$E^*_{(2\text{-}D\,trn)}$	$E^*_{(3\text{-}D\,cub)}$	$E^*_{(3\text{-}D\,fcc)}$
B1	20	2(hp)2(ph2)2(ph)hp2hph	−9	−15	−11	−23
B2	24	2h2p6(hpp)2h	−9	−17	−13	−23
B3	25	pphpphh3(4phh)	−8	−12	−9	−17
B4	36	3phhpphh5p7hpphh4phhpphpp	−14	−24	−18	−38
B5	48	2ph2p2h2p2h5p10h6p2(2h2p)h2p5p	−23	−40	−31	−68
B6	50	2h3(ph)p4hp2(h3p)h4p2(h3p)hp4h3 (ph)p2h	−21	−41	−31	−71
B7	60	2p3hp8h3p10hph3p12h4p6hp2hphp	−36	−70	−54	−128
B8	64	12h2(ph)2(2p2h)2ph2p2(2h2p)2(hpph) hp2(ph)p12h	−42	−75	−58	−128
B9	85	4h4p12h6p12h3p12h3p12h3ph2p2h2p2h2 phph	−53			
B10	100a	6php2h5p3hp5hp2h4p2h2p2hp5hp10hp2h p7h11p7h2php3h6php2h	−48			
B11	100b	3p2h2p4h2p3h2(phh)p4h8p6h2p6h9php 2hp11h2p3hp2hph2php3h6p3h	−50			

Here, E^* gives the optimum fitness for 2-D square ($E^*_{(2\text{-}D\,sqr)}$), 2-D triangular ($E^*_{(2\text{-}D\,trn)}$), 3-D cubic ($E^*_{(3\text{-}D\,cub)}$), and 3-D FCC ($E^*_{(3\text{-}D\,fcc)}$) lattices.

templates. Apart from being similar to the proven technique of template-based modeling, the proposed technique achieves this by gathering knowledge by exploring various basins of attraction. While clustering helps capture knowledge in the form of memes from different basins of attraction, the next challenge is to have a mechanism for combining this information. To achieve this, we implement a two-stage clustering; the overall architecture is shown in Fig. 8.

In stage 1, a number of clusters, κ, are formed with a random initial population set. The individuals are initially generated using the proposed DIG technique described in Section IV-B. Each cluster is allowed to evolve independently until the best fitness value remains unchanged for a specified number of iterations, n_c, and the required number of best individuals are identified, i.e., satisfies (24). Then, using the meme generation technique described in Section IV-D [59], memes are generated for the cluster. At the end of stage 1, we obtain weighted matrix Γ' with memes from each cluster. As we stated earlier, these memes existing within the individuals with higher fitness contain the information of the peaks to which the conformations are converging. The memes from all the clusters containing the information of various basins of attraction are then combined together and used for generating new individuals using the meme-based individual generation process (see Section IV-D2). Let the individuals generated from the memes of each cluster be $\mathbb{P}_\kappa$. If $\mathbb{P}$ is the total number of individuals (the same number of individuals used in clusters of both stage 1 and stage 2), then

$$\mathbb{P}_\kappa = \frac{\mathbb{P}}{\kappa}. \tag{28}$$

Next, the newly generated individuals evolve in the stage-2 cluster using MA with regular local search, crossover, and mutation operations. In the analysis and improvement phase of MA, if the population enters in a local minima [satisfies (24)], then 75% of the worse population are replaced by new individuals generated using DIG.

V. EXPERIMENTS AND RESULTS

The performance of the proposed CMA is validated using known benchmark sequences on different lattice models, namely, 2-D square, 2-D triangular, 3-D cubic, and 3-D FCC. The benchmark sequences for HP model protein [10], [23] are shown in Table II. The proposed CMA is also used for studying the functional model proteins given in [10] and [23], and reproduced in Table III.

For the entire experimentation, the value of constant a occurring in (23) is fixed as 100 to provide the term for topological neighbors with a dominance over the remaining two terms of H-compliance and P-compliance. The population size is maintained at 200 (= $\mathbb{P}$), as in [52], throughout the entire simulation. The mutation is started at a low initial mutation rate of 0.05 ($\mu_0 = 0.05$), as in [52], and the process adapts its value according to (13). To establish the end of stage 1, when the search is deemed to have entered a basin of attraction, we check that the best fitness value does not change for 3 (= n_c) iterations and that most of the individuals of the population have a difference in their fitness value of $\delta = 1.9$ of (24). The memes are calculated from 20 best individuals within a basin of attraction. The number of best individuals are chosen based on the choice from EDA [23] where they used 10% individuals of the population; in our case, 10% of $\mathbb{P}$ is 20. Unless otherwise specified, the number of clusters (κ) used for CMA is 20. Every cluster runs for the number of iterations needed to converge to a basin of attraction [until (24) returns 1]. For populating the stage-2 cluster, an equal number of individuals ($\mathbb{P}/\kappa$) are generated from the memes of each of the primary clusters of stage 1.

A. Effect of Modified Fitness Function

To investigate the impact of our modified fitness function of (23) over the conventional TN fitness function of (1) based only on topological neighbors, we have chosen a large sequence *B9* on a 2-D square lattice. The simulations are carried out using simple genetic algorithm (SGA) with: 1) TN fitness function of (1) and 2) MFF of (23). The comparison is shown in Fig. 9 (also presented in our previous work [59]).

We conducted 20 runs in this experiment. For each of these runs, the termination criterion was a maximum of 450 iterations. For each run, the average fitness of best ten individuals was computed at uniformly distributed iteration counts. After completing all 20 runs, we further averaged these

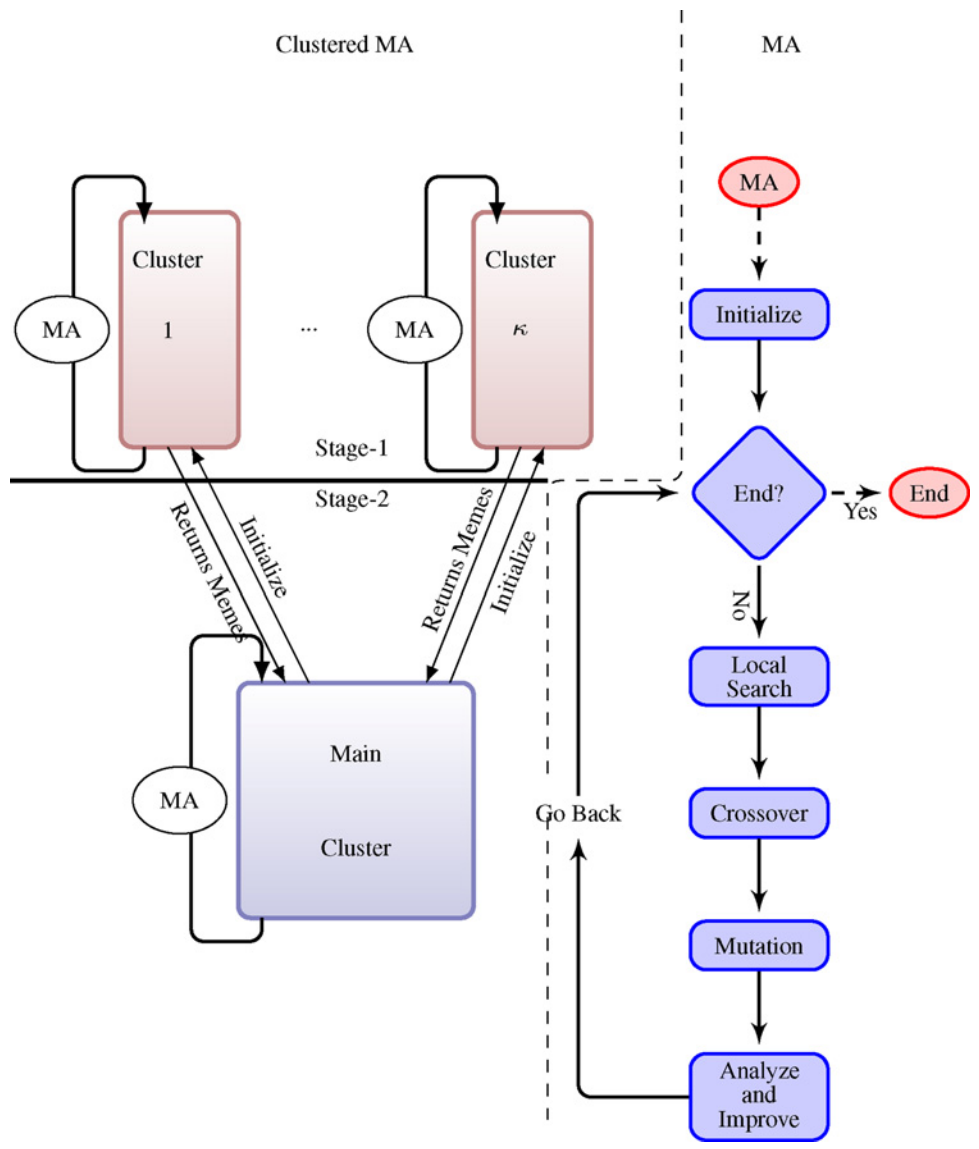


Fig. 8. Schematic of proposed CMA shown on the left side of the vertical dotted line. On the right side of the dotted line, the flow chart of the standard memetic algorithm is shown.

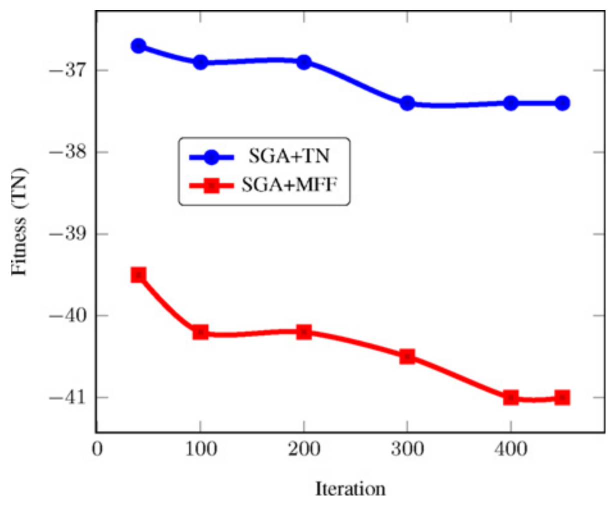


Fig. 9. Comparison of average TN fitness achieved by using standard GA (SGA) with TN function versus standard GA with MFF.

average fitness values at each of these specific iteration counts. The variation of these average values as a function of iteration counts for the two different fitness functions (TN and TFF) is shown in Fig. 9. Clearly, the search with MFF is a much better effect than the search with only a TN function.

B. Effect of Dynamic Generation of Population

For exploring the effect of the proposed DIG method, we specifically choose long sequences (i.e., sequence *B6–B11* from Table II) so that clear distinctions can be observed with a traditional approach. The results of the comparison are shown in Table IV and also in the semi-log plot of Fig. 10. It can be clearly seen that the average time with DIG approach is significantly less than the traditional approach of population generation. The reduced time in generating populations effectively reduces the overall time of proposed CMA and contributes to its superior performance. The DIG approach helps in speeding up the algorithm not only in the

TABLE III
BENCHMARK SEQUENCES FOR FUNCTIONAL MODEL PROTEIN

Seq	Sequence	E*
BF1	PHPPHPPHHHHPPHPPHPHPPHH	−20
BF2	PHPPHPPHHHHHPPPPHPPHPPH	−17
BF3	HPHPHPHHHPPHPPPPHPHHPPHH	−16
BF4	HHHPHHHPPHHPPPHPHPPHHHH	−20
BF5	PHPPPPPPHPHHHPHPHHHHPHPH	−17
BF6	HHPHPPHPPPPHPPPPHPPPHHH	−13
BF7	PHPHHPHHHHHHPPHHHHPHHHHH	−26
BF8	HPHPPPHHHHPHPPPPHPHPHHH	−16
BF9	PHPHHPHHPHHPHPHPHPPPPPH	−15
BF10	HPHPHPPPPPHHPPPHPHPHPHH	−14
BF11	PHPPHHHPHPPHPHHPHPPPPPH	−15

All of the proteins are of equal length of 23. Here, E* gives the optimum fitness in a 2-D square lattice.

TABLE IV
TIME (IN SECONDS) NEEDED FOR GENERATING 200 INDIVIDUALS OF DIFFERENT LENGTHS ON A 2-D SQUARE LATTICE

	Seq	**Run1**	**Run2**	**Run3**	**Run4**	**Avg**
Proposed	B6	0.093	0.078	0.062	0.078	0.077
	B7	0.109	0.093	0.093	0.093	0.097
	B8	0.124	0.124	0.140	0.109	0.124
	B9	0.249	0.187	1.265	0.156	0.464
	B10	0.234	0.296	0.249	0.234	0.253
	B11	1.374	0.874	0.218	12.640	3.776
Traditional	B6	1.609	1.546	1.671	1.718	1.636
	B7	5.843	6.171	6.234	6.593	6.210
	B8	11.343	10.109	11.515	10.109	10.769
	B9	191.123	167.123	191.983	148.264	174.623
	B10	1291.429	1139.570	1104.586	1031.790	1141.844
	B11	1038.524	1044.368	1027.946	971.087	1020.481

initial stage but also in later stages because a small number of new individuals are needed in subsequent iterations due to different operations, such as elitism and twin removal.

C. Effect of Different Lattice Models

To study the effect of lattice models on CMA performance, we consider four well-known lattice models, namely, 2-D square, 3-D cubic, 2-D triangular, and 3-D FCC. For each of the models and *B1–B8* sequences, the simulations are repeated 20 times and the best results with mean and standard deviation are recorded. The best results (E_b) for different lattices are shown in Table V for the benchmark sequence of Table II. For a better understanding of the performance of CMA, we have also included the worst results (E_w) with the mean (μ) and margin of error (γ) for the 10%–90% confidence interval. The confidence interval was calculated using the t-distribution table. For almost all smaller sequences (*B1–B4*), without any exception CMA reaches the optimum (E^*) for all the cases. The results show that the proposed CMA can effectively find minimum fitness values for different lattice models and different benchmark sequences without any parameter optimization. We use a single set of parameters and all the necessary adaptations are done automatically within the proposed technique.

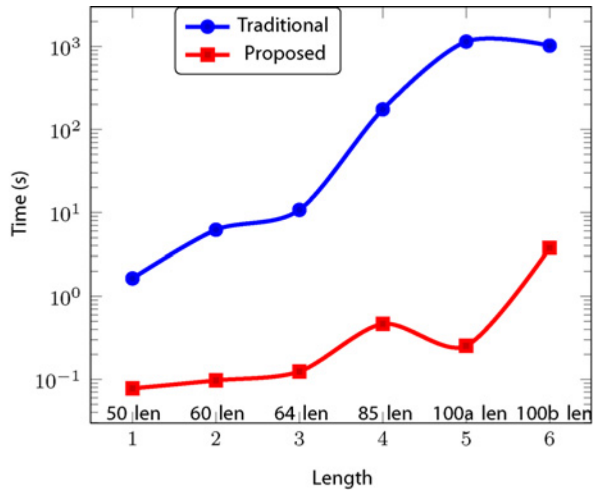


Fig. 10. Semi-log plot for comparison between average time (in seconds) for generating individuals using traditional (random) approach and the proposed (dynamic) approach.

D. Effect of Memes

The proposed meme identification and replacement technique gives us two advantages. First, it can preserve multiple peaks (an efficient niching technique) of the multimodal search space of the PSP problem. Second, it makes the process faster. To substantiate experimentally, we randomly choose sequence *B5* on the 3-D cubic lattice model. The results, one without using meme information and the other using meme information, are shown in Tables VI and VII, respectively. First, we carry out five runs of simulations without using meme identification and meme replacement techniques. For each run, the maximum iteration count in stage 1 was set to 1000. The average fitness was found to be equal to −27. In the second experiment for the meme generation technique, we randomly chose five stage-1 clusters. While the maximum iteration count of stage 1 was maintained at 1000, the maximum iteration count of stage 2 was set to 150. But, as explained earlier, the stage-1 clusters terminate using (24). The average fitness of the top ten individuals is equal to −28.8. To have a similar comparison, we clarify that, in CMA, we run the whole process sequentially (i.e., in stage 1, one cluster at a time) and aggregate the time needed to complete the individual simulation runs. Apart from the improvement in the average fitness value, Tables VI and VII show that use of meme information also reduces the required number of functional evaluations (number of times a fitness function has been evaluated).

E. Effect of Pull Moves

The different pull move scenarios described in Section IV-E exploit the surrounding neighbors more efficiently, compared to Lesh *et al.*'s pull move [53]. A comparison on the number of individuals found by applying different pull moves is given in Table VIII. The different pull moves help us to explore individuals in the neighbor of a individual and also improve the fitness of the individual by choosing the best one from its neighbor.

TABLE V
BEST RESULT ACHIEVED BY CMA FOR DIFFERENT LATTICE MODELS

Seq	**2-D Square**			**3-D Cubic**			**2-D Triangular**			**3-D FCC**		
	$\mathbf{E}_b$	$\mu \pm \gamma$	$\mathbf{E}_w$	$\mathbf{E}_b$	$\mu \pm \gamma$	$\mathbf{E}_w$	$\mathbf{E}_b$	$\mu \pm \gamma$	$\mathbf{E}_w$	$\mathbf{E}_b$	$\mu \pm \gamma$	$\mathbf{E}_w$
B1	−9	−9 ± 0.00	−9	−11	−11 ± 0.00	−11	−15	−15 ± 0.00	−15	−23	−23 ± 0.00	−23
B2	−9	−9 ± 0.00	−9	−13	−13 ± 0.00	−13	−17	−17 ± 0.00	−17	−23	−23 ± 0.00	−23
B3	−8	−8 ± 0.00	−8	−9	−9 ± 0.00	−9	−12	−12 ± 0.00	−12	−17	−17 ± 0.00	−17
B4	−14	−14 ± 0.00	−14	−18	−18 ± 0.00	−18	−24	−24 ± 0.00	−24	−38	−38 ± 0.00	−38
B5	−23	−22.87 ± 0.10	−22	−31	−31 ± 0.00	−31	−40	−40 ± 0.00	−40	−68	−66.94 ± 0.28	−65
B6	−21	−21 ± 0.00	−21	−31	−31 ± 0.00	−31	−41	−40.35 ± 0.15	−40	−71	−69.15 ± 0.37	−67
B7	−36	−35.1 ± 0.09	−35	−54	−52.52 ± 0.20	−51	−70	−70 ± 0.00	−70	−128	−125.65 ± 0.34	−124
B8	−42	−39.6 ± 0.26	−39	−58	−56.3 ± 0.35	−55	−75	72.5 ± 0.34	−70	−128	−124.67 ± 0.45	−123

Here, E_b is the best fitness, E_w is the worst fitness, μ is the mean, and γ is the margin of error for 10%−90% confidence interval.

TABLE VI
NUMBER OF FUNCTIONAL EVALUATIONS (FE) WITH BEST AND AVERAGE (OF TOP TEN INDIVIDUALS) FITNESS VALUES WITHOUT APPLYING MEMES FOR BENCHMARK SEQUENCE **B5**

Without Meme	**Best**	**Avg**	**FE**
1	−27	−27	200 200
2	−29	−29	200 200
3	−25	−25	200 200
4	−28	−28	200 200
5	−26	−26	200 200
Avg	**−27**	**−27**	**200 200**

TABLE VII
NUMBER OF FUNCTIONAL EVALUATIONS (FE) WITH BEST AND AVERAGE (OF TOP TEN INDIVIDUALS) FITNESS VALUES APPLYING MEMES FOR BENCHMARK SEQUENCE **B5**

With Meme	**Best**	**Avg**	**FE**
1	−29	−29	38 200
2	−29	−29	38 000
3	−29	−29	38 800
4	−28	−28	39 200
5	−29	−29	38 800
Avg	**−28.8**	**−28.8**	**38 600**

Five randomly chosen clusters were considered for each run.

F. Effect of Two-Stage Implementation

To understand the effect of two-stage implementation of the algorithm, let us compare the results presented in Table VI for single-stage implementation without memes and those in Table VII for the two-stage approach with memes. In both cases, we have randomly selected the benchmark sequence *B5* for studies. While the single-stage (single-cluster) approach results in an average fitness value of −27 with 1000 iterations, the two-stage approach gives a better fitness value of −28.8 within a much smaller number of iterations, 187.

Stage-1 clusters identify different BOA and pass the domain-specific knowledge via memes to stage-2 cluster. In stage 1, without allowing full convergence, a minimal number of iterations are carried out to reach the near-convergence condition. In the stage-2 cluster, using the meme-based individuals, the simulation is run until a convergence is achieved. Again, to empirically show the effect of a meme on the stage-2 cluster, we have carried out an experiment where four stage-1 clusters were considered for a population of 200. Memes generated from the stage-1 clusters are passed to the stage-2 cluster and, there, evolution is done utilizing these memes. The results of five simulation runs for seven benchmark sequences (i.e., *B5–B11*) are shown in Table IX. Note that for sequence B5, the optimum fitness value is achieved in cluster 2 of stage 1 for run 1, run 2, and run 4. Hence, runs for neither cluster 3, cluster 4 of stage 1, or the final run of stage 2 are required. From the result, it is evident that using only these memes, stage 2 can quickly converge to a better solution of all the solutions found in stage 1 for different basins of attraction.

G. Effect of the Number of Clusters

Let us now investigate the effect of varying the number of clusters on the CMA performance using the known benchmark sequences *B*5, *B*7, *B*8, *B*9, *B*10, and *B*11. The population size is $\mathbb{P}$ = 200. As explained earlier, the population in a stage-2 cluster is based on the memes generated in the stage-1 clusters. For simulation experiments, five runs each are carried out for 12 h on two different grids [Victorian Partnership for Advanced Computing (VPAC)[1] and Monash Campus Grid (MCG)[2]]. However, rather than using these as clusters, each machine is run independently for independent benchmark sequences. Thus, the computation time for a sequence is equivalent to time required for running it on a single machine. The average fitness values obtained from these studies are shown in Table X (for better visualization we have eliminated negative signs) and Fig. 11. In Fig. 11(b) and (d), although a slight dip in fitness for four, eight, and ten clusters is observed, we note from the fitness values of Table X for these clusters that the variation for both B7 and B9 is very small. As explained in Section IV-F, such small-magnitude dips can occur during search since the convergence of each cluster depends on many factors, e.g., size and number of basins of attraction, nature of primary sequence. We note that the performance of CMA improves with the increase in the number of clusters applied. This is in agreement with (26), which shows that an increase in clusters will increase the probability of finding different peaks in the search space despite the effect of genetic drift. However, it should be noted

[1]Available at http://www.vpac.org.
[2]Available at http://www.monash.edu.au/eresearch/services/mcg/index.html.

TABLE VIII
EFFECT OF PULL MOVES IS SHOWN IN TERMS OF NUMBER OF NEIGHBORS FOUND FOR DIFFERENT SCENARIOS OF PULL MOVES ON $B5$ ON A 3-D CUBIC LATTICE

	Initial Random Individual	Forward Pull		Reverse Pull		Begin Pull		End Pull	
# of Runs	TN	# found	TN	# found	TN	# found	TN	# found	TN
1	0	121	−3	240	−4	249	−4	258	−4
2	−3	121	−5	241	−6	246	−6	254	−6
3	−12	103	−13	204	−13	212	−13	221	−13
4	−5	102	−7	204	−8	213	−8	221	−8
5	−8	93	−10	185	−10	193	−10	202	−10
6	−1	123	−3	247	−5	256	−5	264	−5
7	−1	129	−3	256	−3	265	−6	274	−6
8	0	139	−2	279	−3	288	−3	297	−3
9	−2	134	−5	268	−5	276	−5	283	−5
10	−4	98	−8	192	−8	198	−8	207	−8

Only first iteration run is considered.

TABLE IX
FITNESS VARIATION IN STAGE 1 AND STAGE 2 FOR EACH OF THE FOUR CLUSTERS SHOWN

		Stage 1				Stage 2
Seq	Run	Cluster-1	Cluster-2	Cluster-3	Cluster-4	Final
	1	−21(7)	−23(2)	–	–	–
	2	−20(5)	−23(5)	–	–	–
B5	3	−21(5)	−21(7)	−20(4)	−21(4)	−22(5)
	4	−20(5)	−23(5)	–	–	–
	5	−21(6)	−21(6)	−22(8)	−20(4)	−22(1)
	1	−33(3)	−34(5)	−33(5)	−33(5)	−35(23)
	2	−33(4)	−34(5)	−33(5)	−34(6)	−35(4)
B7	3	−34(8)	−33(4)	−32(5)	−34(8)	−35(0)
	4	−33(5)	−34(6)	−34(5)	−34(7)	−35(704)
	5	−33(4)	−33(5)	−35(6)	−32(5)	−36(4)
	1	−38(8)	−34(4)	−35(7)	−36(7)	−38(0)
	2	−37(10)	−38(5)	−37(8)	−36(6)	−39(825)
B8	3	−37(7)	−35(6)	−36(8)	−35(6)	−38(884)
	4	−39(12)	−36(6)	−37(10)	−35(5)	−39(0)
	5	−37(8)	−34(6)	−35(5)	−38(8)	−40(22)
	1	−46(5)	−48(6)	−48(7)	−49(8)	−50(101)
	2	−50(8)	−47(8)	−46(6)	−47(8)	−51(17)
B9	3	−47(9)	−47(6)	−46(7)	−46(7)	−49(149)
	4	−45(4)	−48(8)	−46(5)	−49(7)	−51(388)
	5	−47(6)	−45(6)	−46(6)	−48(6)	−52(179)
	1	−41(5)	−43(5)	−45(7)	−43(8)	−46(13)
	2	−41(6)	−43(8)	−41(7)	−42(7)	−45(95)
B10	3	−45(8)	−42(6)	−42(6)	−42(7)	−46(15)
	4	−42(7)	−42(7)	−42(6)	−44(8)	−44(1)
	5	−40(5)	−43(6)	−41(6)	−42(7)	−45(39)
	1	−40(3)	−41(7)	−44(7)	−42(5)	−46(78)
	2	−45(8)	−44(7)	−44(6)	−42(5)	−47(2)
B11	3	−43(8)	−43(8)	−42(8)	−42(9)	−44(1)
	4	−44(7)	−42(5)	−42(6)	−40(5)	−47(19)
	5	−40(5)	−42(13)	−42(7)	−42(6)	−46(92)

Numbers in parentheses indicate the number of iterations required. A "—" indicates the optimum is reached earlier. Iteration count "0" indicates that at the time of initialization the fitness was achieved.

TABLE X
AVERAGE FITNESS WITH A DIFFERENT NUMBER OF CLUSTERS FOR BENCHMARK SEQUENCES USING 2-D SQUARE LATTICE

#Cluster	B5	B7	B8	B9	B10	B11
1	22.0	34.3	37.8	48.6	43.3	45.9
2	22.2	34.7	38.7	50.1	44.1	45.9
4	22.3	35.2	38.9	50.4	44.9	46.0
8	22.6	35.1	38.9	50.3	45.2	46.4
10	22.7	35.0	39.5	50.2	45.3	47.1
20	22.9	35.4	39.5	50.7	45.7	47.2

that any increase in accuracy due to an increase in the number of clusters comes at the cost of increased computational time. Further, we also observe that the degree of improvement is not linear but reduces as the number of clusters increases. Moreover, as expected, the impact of the number of clusters depends on benchmark sequences. In fact, if the choice of the number of clusters is equal to the number of basins of attraction, it would lead to an optimal condition of improved accuracy and minimal computation time. However, this is difficult to achieve since the number of basins of attraction for any sequence are not known *a priori*.

H. Comparison of CMA With Other Methods

1) *2-D Square Lattice Model:* To compare CMA's performance on a 2-D square lattice model, we consider five recent approaches, namely, IA [10], NewACO [18], guided genetic algorithm (GGA) [52], EDA [23], and PERM [60]. The studies are carried out using benchmark sequences with the 2-D square HP lattice model. The CMA is designed with a population size of $\mathbb{P} = 200$ generated using the DIG technique and the number of stage-1 clusters set as $\kappa = 20$. The adaptive mutation rate μ_t is computed dynamically using (13). It uses the meme generation technique described in Section IV-D to capture the domain knowledge of the 20 basins of attraction and, as explained earlier, subsequently to generate the population using memes for a stage-2 cluster. For the population of a stage-2 cluster, memes from each of the 20 clusters are used to generate 10 ($= \mathbb{P}/\kappa$) individuals. Thus, a total of 200 individuals from memes of 20 clusters are obtained. Like EDA, 50 trial runs are taken for CMA for each of the benchmark sequences (see Table II) and the best results achieved for CMA along with other EAs are shown in Table XI.

Although EDA shows a similar performance to CMA, it is achieved using a population size of 5000, which is 50 times larger than that applied for CMA. Moreover, CMA outperforms EDA and NewACO for the two benchmark sequences,

TABLE XI
COMPARISON OF RESULTS ACHIEVED BY DIFFERENT SEARCH ALGORITHMS ON 2-D SQUARE LATTICE FOR THE BENCHMARK SEQUENCES OF TABLE II

Seq	IA [10]	New ACO [18]	EDA [23]	GGA [52]	PERM [60]	CMA	$\mu \pm \gamma$
B1	−9	−9	−9	−9	−9	−9	−9 ± 0.00
B2	−9	−9	−9	−9	−9	−9	−9 ± 0.00
B3	−8	−8	−8	−8	−8	−8	−8 ± 0.00
B4	−14	−14	−14	−14	−14	−14	−14 ± 0.00
B5	−23	−23	−23	−23	−23	−23	−22.87 ± 0.06
B6	−21	−21	−21	−21	−21	−21	−21 ± 0.00
B7	−35	−36	−35	−36	−36	−36	−35.04 ± 0.06
B8	−42	−42	−42	−42	−38	−42	−39.41 ± 0.14
B9		−51	−52		−53	−53	−50.91 ± 0.17
B10		−47	−47		−48	−47	−45.41 ± 0.20
B11		−47	−48		−50	−50	−47.29 ± 0.18

Mean (μ) and margin of error (γ) for 10%−90% confidence interval for CMA.

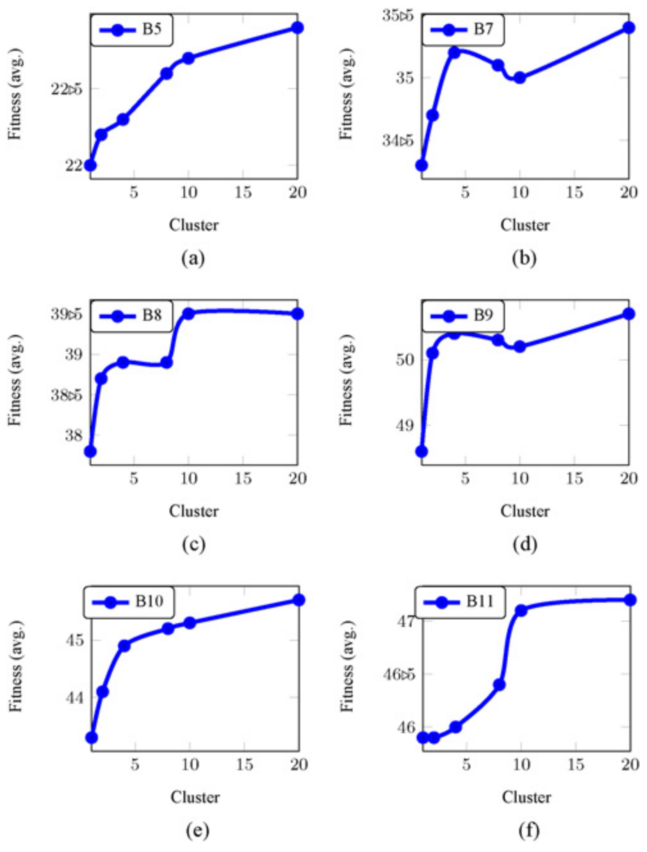


Fig. 11. Effects of the number of clusters on fitness for different benchmark sequences for (a) *B*5, (b) *B7*, (c) *B8*, (d) *B9*, (f) *B10*, and (e) *B11*).

i.e., *B9* (NewACO = −51, EDA = −52, and CMA = −53) and *B11* (NewACO = −47, EDA = −48, and CMA = −50). PERM [60] finds the best results for *B*9, *B*10, and *B*11 with a number of runs required being 20–200. PERM performs well in all cases but less consistently as it underperforms for *B*8 with other methods including CMA. The results clearly show that CMA performs consistently and achieves high fitness values with a smaller population size. Note that CMA studies also include the mean (μ) and margin of error (γ) for the 10%–90% confidence interval. As these values are not available for other algorithms, no comparison could be made. Further, it can be noted that the improvements in performance become more significant for more complex protein models. This is demonstrated with the analysis of the 3-D cubic lattice model carried out next. Comparisons with other approaches, such GA [11], MMA [29], and other EAs [8], [61], [62], are not included here because either their performance was inferior to CMA or the number of iterations required to obtain optimal fitness was much higher.

TABLE XII
COMPARISON OF RESULTS ACHIEVED BY DIFFERENT SEARCH ALGORITHMS ON 3-D CUBIC LATTICE FOR THE BENCHMARK SEQUENCES OF TABLE II

Seq	BestHGA [61]	BestIM [10]	BestEDA [23]	BestCMA
B1	−11	−11	−11	−11
B2	−13	−13	−13	−13
B3	−9	−9	−9	−9
B4	−18	−18	−18	−18
B5	−28	−29	−29	−31
B6	−26	−23	−31	−31
B7	−49	−41	−49	−54
B8	−46	−42	−52	−58

2) *3-D Cubic Lattice Model:* In this experiment, we compare the performance of CMA with hybrid GA [61], IA [10], and EDA [23]. With the same number of initial population ($\mathbb{P}$ = 200) and with 20 stage-1 clusters (κ = 20), the best value from 20 simulation runs is shown in Table XII. As in previous experiments, we have considered the time limit as the termination criteria. Each simulation was run for 12 h on the two clusters VPAC and MCG. As mentioned in Section V-G, the machines in the clusters are used independently and not as a cluster. The best result for EDA and hybrid GA was obtained using a population size of 5000; the number of iterations for EDA was 1000 whereas for hybrid GA it was 5×10^6. Not only does our proposed CMA show superior performance compared to other approaches (see Table XII) but it also achieves this by maintaining a constant population size of only $\mathbb{P}$ = 200 for all instances.

3) *Functional Model Protein:* Similar to two recently reported approaches in this journal [10], [23] and for the sake of comparison, we also present analysis with the functional model protein for the same set of benchmark sequences (see Table III) in a 2-D square lattice. Two different techniques exist in the literature to measure the performance of different algorithms: 1) number of iterations required for the best run to reach the

TABLE XIII
COMPARISON OF RESULTS ACHIEVED BY DIFFERENT SEARCH ALGORITHMS FOR FUNCTIONAL MODEL PROTEINS OF TABLE III ON 2-D SQUARE LATTICE

	BestIM [10]			BestEDA [23]			BestCMA		
Seq	S	$\bar{f}$	Algo.	S	$\bar{f}$	Algo.	S	$\bar{f}$	$\bar{t}$(h:mm:ss)
BF1	100	32647.73(3372)	AI_2	100	25799.3(12 650)	EDA_1	100	9.8(2)	0:00:08.852
BF2	100	17526.73(578)	AI_2	100	20593.1(2750)	EDA_1	100	13.2(2)	0:00:11.930
BF3	3.33	1194332(100 234)	AI_1	100	140 802(35 900)	EDA_1	80	12401.1(10 514)	5:12:27.190
BF4	100	128015.1(4955)	AI_2	100	108 118(15 900)	EDA_1	100	351.2(4)	0:06:54.285
BF5	100	12059.33(1047)	AI_2	100	197 156(20 950)	EDA_3	100	19.6(10)	0:00:22.999
BF6	100	2732830(2828)	AI_2	100	61577.6(5420)	EDA_2	100	29.4(7)	0:00:24.371
BF7	100	584179.8(10 061)	AI_2	80	3 433 060(19 450)	EDA_3	100	116(23)	0:02:33.120
BF8	100	38262.6(1818)	AI_2	100	147 938(10 350)	EDA_2	100	9.4(3)	0:00:10.268
BF9	100	281720.8(3845)	AI_2	100	155 722(4950)	EDA_3	100	13.4(3)	0:00:18.282
BF10	100	100085.43(2847)	AI_3	100	57652.1(8950)	EDA_2	100	93.2(1)	0:01:54.828
BF11	100	27743.7(1007)	AI_2	100	11927.1(2950)	EDA_2	100	3.2(1)	0:00:03.730

Numbers within parentheses indicate the best generation count. Here, S is success rate and $\bar{t}$ is average time needed for CMA and $\bar{f}$ is for average iteration count. AI_1: IA elitism aging, AI_2: IA pure aging, AI_3: IA memory cell, EDA_1: MK-EDA_2, EDA_2: tree-EDA, and EDA_3: MT-EDA_4.

optimum solution [29] and 2) success rate with the average number of iterations required to reach the optimum [10]. We present both of these along with the average time needed to show the superiority of CMA over the other two techniques.

IA [10] has been tested with its three variants: IA with elitist aging, IA with pure aging, and IA using memory cells ($\tau_B = 5$, $\tau_{B_{mem}} = 10$). We consider the best of the three results for comparison. In our experiment, we ran CMA with a population size of $\mathbb{P} = 200$ and the number of clusters in stage-1 κ as 8. Along with the average generation count, the best generation count is shown in the parentheses. The comparison of the results is shown in Table XIII. Since CMA incorporates a local improvement process, its iteration count for any generation may not be comparable to EDA or IA. Hence, we also determine the time needed to reach the optimal solution. The average time of CMA was taken from five runs and the maximum time limit was kept as 12 h. EDA [23] gets a result for functional model protein by discarding, in each evolution, all except the best individuals and generates new individuals to replace those that are discarded. In this case, they consider a population size $\mathbb{P} = 500$, allowing for a minimum number of different individuals to be 50. EDA is based essentially on the distribution of the best individual set and due to the absence of a crossover operation; the individuals do not get any chance to share their information. It is more likely that the new individuals generated in this way may fail to meet SAW constraints, thereby increasing the time complexity. Further, inserting too many new individuals can shift the search space randomly from one basin of attraction to another. Moreover, with the absence of local search, proper exploration of a basin of attraction may not take place, resulting in slowing down the whole convergence process.

From the results, we can easily observe that CMA has consistently performed better than the other two approaches. The elevated complexity in convergence of sequence *BF3*, indicated by a high average evolution count $\bar{f}$ and low success rate S of IA is in concurrence with the corresponding values (along with an associated longer convergence time) obtained by the proposed CMA. While the two variants of EDA, namely, tree-EDA and MT-EDA_4, in [23], respectively, show a significantly high value of average evolution count of 997 618 and 835 909, the reasons for the comparatively low value for its third variant MK-EDA_2 (shown in the table here) were not reported. As EDA involves the generation of a whole population in each iteration, it can consume significant time as we theoretically and experimentally showed that random individual generation is very computationally expensive [50]. Although the individual generation in EDA is not the same as the random individual generation technique, the individuals generated from the sampled probability distribution do not guarantee an SAW that is the same as the random individual generation technique discussed in [50].

A similar analysis cannot be carried out for IA because it involves multiple mutation operations in every generation depending on a function called mutation potential, M, and it also generates new individuals by birth phase when the selection cannot find d B cells from the three set of populations $P^{(t)}$, $P^{(hyp)}$, and $P^{(macro)}$ [10]. The hypermutation operation of IA will randomly shift the individual to any unknown location on the search space, while hypermacromutation can be considered a kind of neighborhood search (local search) technique that is essentially a fully random search without using any problem-specific domain information. These random operations will slow down the convergence process of IA, which can easily be observed by the empirical results of EDA [23].

VI. CONCLUSION

In this paper, we proposed a two-stage novel framework for implementing MA for protein structure prediction using low-resolution lattice models. The combined global and local search features of MA emulated the nature's folding principles, i.e., local optimization first followed by global optimization. The proposed MA has various novel features, i.e., MFF capturing the physico-chemical properties of the residues, DIG, generation of new conformations based on concept of meme, applications of pull moves for local optimization, and clustering of a population to cope with the multimodal search difficulties. MFF took into account the effect of hydrophobicity and hydrophilicity of the amino acids and made the fitness function take into account the behavior of folding taking place in Nature. Dynamic generation of individuals prevented the generations of invalid (non-SAW) conformations. The meme-based generation of individuals was implicitly equivalent to the concept of substructures-based protein modeling applied in a homology approach. The memes, which are one of the key strengths in the operation of this algorithm, identified the domain knowledge (i.e., the existence of substructures at specified locations of conformation) within a particular basin of attraction. In other words, the meme's identification and its application were akin to heuristically identifying protein substructures corresponding to a specific basin of attraction. Various scenarios of pull moves helped in quickly finding out highly fit individuals in its neighbor. Clusters contributed to dealing with the multimodal nature of the PSP problem and gave an effective niching technique to deal with multiple peaks of the search space. Thus, the entire gamut of novel features made the proposed CMA efficient and robust in solving the low-resolution PSP problem.

The proposed clustered memetic algorithm is generic because it is able to determine the optimum structures from various types of lattice models without the need to change the initial parameters. Other approaches found in the literature need different sets of parameters for different lattice models. Furthermore, the approach is dynamic since the parameters get updated automatically during generations based on the individuals. The meme identification and meme transfer within the clustered memetic algorithm is also generic and portable as it can be applied to any multimodal NP-complete problem (such as job shop scheduling, bin packing, or the traveling salesman problem). For this, we can simply cast these problems into MA formalism and apply the clustered MA approach by identifying memes and transferring them to a stage-2 cluster. As Goldberg *et al.* [49] stated, if the search space can be divided into different types of basins of attraction, it implies that it would be possible to identify common substructures (schema) in the GA representations for all of these types of problems.

Like all stochastic search techniques, the proposed technique has the inherent capability of parallelization. This means that, though we ran it sequentially on a single machine, it can easily be implemented into a grid or cloud infrastructure for higher resolution models that form the next stage of the overall hierarchical PSP approach.

REFERENCES

[1] D. Kennedy and C. Norman, "Editorial: So much more to know," *Science*, vol. 309, no. 5731, pp. 78–102, 2005.

[2] J. Berg, J. Tymoczko, and L. Stryer, *Biochemistry*. San Francisco, CA: Freeman, 2002.

[3] B. Berger and T. Leighton, "Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete," *J. Computat. Biol.*, vol. 5, no. 1, pp. 27–40, 1998.

[4] W. E. Hart and S. C. Istrail, "Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal," *J. Computat. Biol.*, vol. 3, no. 1, pp. 53–96, 1996.

[5] G. Mauri, G. Pavesi, and A. Piccolboni, "Approximation algorithms for protein folding prediction," in *Proc. 10th Annu. ACMSIAM Symp. Discrete Algorithms*, 1999, pp. 945–946.

[6] U. H. Hansmann and Y. Okamoto, "New Monte Carlo algorithms for protein folding," *Current Opinion Structural Biol.*, vol. 9, no. 1, pp. 177–183, 1999.

[7] C. Thachuk, A. Shmygelska, and H. H. Hoosr, "A replica exchange Monte Carlo algorithm for protein folding in the HP model," *BMC Bioinformatics*, vol. 8, no. 342, 2007.

[8] V. Cutello, G. Nicosia, and M. Pavone, "An immune algorithm with hyper-macromutations for the Dill's 2-D hydrophobic-hydrophilic model," in *Proc. IEEE CEC*, Jun. 2004, pp. 1074–1080.

[9] V. Cutello, G. Morelli, G. Nicosia, and M. Pavone, "Evolutionary computation in combinatorial optimization," in *Immune Algorithms With Aging Operators for the String Folding Problem and the Protein Folding Problem* (Lecture Notes in Computer Science). Berlin/Heidelberg, Germany: Springer-Verlag, 2005, pp. 80–90.

[10] 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, Feb. 2007.

[11] R. Unger and J. Moult, "Genetic algorithms for protein folding simulations," *J. Molecular Biol.*, vol. 231, no. 1, pp. 75–81, 1993.

[12] A. L. Patton, W. F. Punch, III, and E. D. Goodman, "A standard GA approach to native protein conformation prediction," in *Proc. 6th Int. Conf. Genetic Algorithms*, 1995, pp. 574–581.

[13] M. M. Khimasia and P. V. Coveney, "Protein structure prediction as a hard optimization problem: The genetic algorithm approach," *Molecular Simulation*, vol. 19, no. 4, pp. 205–226, 1997.

[14] N. Krasnogor, W. E. Hart, J. Smith, and D. A. Pelta, "Protein structure prediction with evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, 1999, pp. 1596–1601.

[15] N. Krasnogor, D. Pelta, P. E. M. Lopez, and E. de la Canal, "Genetic algorithms for the protein folding problem: A critical view," in *Proc. Eng. Intell. Syst.*, 1998.

[16] M. T. Hoque, M. Chetty, and L. S. Dooley, *A Hybrid Genetic Algorithm for 2-D FCC Hydrophobic-Hydrophilic Lattice Model to Predict Protein Folding* (Lecture Notes in Computer Science, vol. 4304). Berlin/Heidelberg, Germany: Springer-Verlag, 2006, pp. 867–876.

[17] M. Hoque, M. Chetty, and A. Sattar, "Protein folding prediction in 3-D FCC HP lattice model using genetic algorithm," in *Proc. IEEE CEC*, Sep. 2007, pp. 4138–4145.

[18] A. Shmygelska and H. H. Hoos, "Advances in artificial intelligence," in *An Improved Ant Colony Optimization Algorithm for the 2-D HP Protein Folding Problem* (Lecture Notes in Computer Science, vol. 2671). Berlin/Heidelberg, Germany: Springer-Verlag, 2003, p. 993.

[19] A. Shmygelska and H. H. Hoos, "An ant colony optimization algorithm for the 2-D and 3-D hydrophobic polar protein folding problem," *BMC Bioinformatics*, vol. 6, no. 30, 2005.

[20] X.-M. Hu, J. Zhang, and Y. Li, "Computational intelligence in biomedicine and bioinformatics," in *Flexible Protein Folding by Ant Colony Optimization* (Studies in Computational Intelligence, vol. 151). Berlin/Heidelberg, Germany: Springer-Verlag, 2008, pp. 317–336.

[21] T. Thalheim, D. Merkle, and M. Middendorf, "Protein folding in the HP-model solved with a hybrid population based ACO algorithm," *IAENG Int. J. Comput. Sci.*, vol. 35, no. 3, pp. 291–300, 2008.

[22] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in 2-dimensional lattices with estimation of distribution algorithms," in *Proc. 1st Int. Symp. Biol. Med. Data Anal.* (Lecture Notes in Computer Science, vol. 3337). Barcelona, Spain: Springer-Verlag, 2004, pp. 388–398.

[23] 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, Aug. 2008.

[24] C. Levinthal, "Are there pathaways for protein folding?" *J. Chim. Phys. Phys. Chim. Biol.*, vol. 65, pp. 44–45, 1968.

[25] K. A. Dill, S. B. Ozkan, T. R. Weikl, J. D. Chodera, and V. A. Voelz, "The protein folding problem: When will it be solved," *Current Opinion Structural Biol.*, vol. 17, no. 3, pp. 342–346, 2007.

[26] A. Fallahi, C. Prins, and R. W. Calvo, "A memetic algorithm and a tabu search for the multicompartment vehicle routing problem," *Comput. Oper. Res.*, vol. 35, no. 5, pp. 1725–1741, 2008.

[27] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 62–69, Feb. 2007.

[28] A. Caponio, L. Cascella, and F. Neri, "A fast adaptive memetic algorithm for online and offline control design of PMSM drives," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 28–41, Feb. 2007.

[29] N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst, "Multimeme algorithms for protein structure prediction," in *Proc. Parallel Problem Solving From Nature (PPSN VII)*, LNCS 2439. 2002, pp. 769–778.

[30] A. Bazzoli and A. G. Tettamanzi, "A memetic algorithm for protein structure prediction in a 3-D lattice HP model," in *Applications of Evolutionary Computing* (Lecture Notes in Computer Science, vol. 3005). Berlin/Heidelberg, Germany: Springer, 2004, pp. 1–10.

[31] D. Pelta and N. Krasnogor, "Multimeme algorithms using fuzzy logic based MEMES for protein structure prediction," in *Recent Advances*

in Memetic Algorithms (Studies in Fuzziness and Soft Computing, vol. 166), W. Hart, J. Smith, and N. Krasnogor, Eds. Berlin/Heidelberg, Germany: Springer, 2005, pp. 49–64.

[32] J. Smith, "The co-evolution of memetic algorithms for protein structure prediction," in *Recent Advances in Memetic Algorithms* (Studies in Fuzziness and Soft Computing, vol. 166), W. Hart, J. Smith, and N. Krasnogor, Eds. Berlin/Heidelberg, Germany: Springer, 2005, pp. 105–128.

[33] C. L. Pierri, A. D. Grassi, and A. Turi, "Lattices for ab initio protein structure prediction," *Proteins Structure Function Bioinformatics*, vol. 73, no. 2, pp. 351–361, 2008.

[34] K. F. Lau and K. A. Dill, "A lattice statistical mechanics model of the conformational and sequence spaces of proteins," *Macromolecules*, vol. 22, no. 10, pp. 3986–3997, 1989.

[35] R. Samudrala, Y. Xia, and E. Huang, "Ab initio protein structure prediction using a combined hierarchical approach," *Proteins Structure Function Bioinformatics*, vol. 37, no. 3, pp. 194–198, 1999.

[36] Y. Xia, E. S. Huang, M. Levitt, and R. Samudrala, "Ab initio construction of protein tertiary structures using a hierarchical approach," *J. Molecular Biol.*, vol. 300, no. 1, pp. 171–185, 2000.

[37] M. Mann, R. Backofen, and S. Will, "Equivalence classes of optimal structures in HP protein models including side chains," in *Proc. Workshop Constraint Based Methods Bioinformatics*, 2009, pp. 43–50.

[38] M. Mann, M. A. Hamra, K. Steinhöfel, and R. Backofen, "Constraintbased local move definitions for lattice protein models including side chains," in *Proc. Workshop Constraint Based Methods Bioinformatics*, 2009, pp. 51–59.

[39] T. Lazaridis and M. Karplus, "Effective energy function for proteins in solution," *Current Opinion Structural Biol.*, vol. 10, no. 2, pp. 139–145, 2000.

[40] C. Hardin, T. V. Pogorelov, and Z. Luthey-Schulten, "Ab initio protein structure prediction," *Current Opinion Structural Biol.*, vol. 12, no. 2, pp. 176–181, 2002.

[41] G. W. GreenWood and J.-M. Shin, *On the Evolutionary Search for Solutions to the Protein Folding Problem*. San Francisco, CA/Oxford, U.K.: Morgan Kaufmann/Elsevier Science, 2003, ch. 6, p. 115.

[42] T. Hoque, M. Chetty, and L. S. Dooley, "Nonisomorphic coding in lattice model and its impact for protein folding prediction using genetic algorithm," in *Proc. IEEE Symp. Comput. Intell. Bioinformatics Comput. Biol.*, 2006, pp. 28–29.

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

[44] P. Moscato, C. Cotta, and A. Mendes, *Memetic Algorithm* (New Optimization Techniques in Engineering). Berlin/Heidelberg, Germany: Springer-Verlag, 2004, ch. 3, pp. 54–85.

[45] R. Dawkins, "Memes: The new replicators," in *The Selfish Gene*. Oxford, U.K.: Oxford Univ. Press, 1976, ch. 11, pp. 203–215.

[46] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992.

[47] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan, 1975.

[48] M. T. Hoque, M. Chetty, and L. S. Dooley, "Pattern recognition in bioinformatics," in *Generalized Schemata Theorem Incorporating Twin Removal for Protein Structure Prediction* (Lecture Notes in Computer Science). Berlin/Heidelberg, Germany: Springer-Verlag, 2007, pp. 84–97.

[49] D. E. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in *Proc. Genet. Evol. Comput. Conf.*, 1999, pp. 220–228.

[50] M. K. Islam and M. Chetty, "Clustered memetic algorithm for protein structure prediction," in *Proc. IEEE CEC*, Jul. 2010.

[51] Y. Zhang, "Progress and challenges in protein structure prediction," *Current Opinion Structural Biol.*, vol. 18, no. 3, pp. 342–348, 2008.

[52] M. C. M. Tamjidul Hoque and A. Sattar, "Biomedical data and applications," in *Genetic Algorithm in Ab Initio Protein Structure Prediction Using Low Resolution Model: A Review* (Studies in Computational Intelligence, vol. 224). Berlin/Heidelberg, Germany: Springer-Verlag, 2009, pp. 317–342.

[53] N. Lesh, M. Mitzenmacher, and S. Whitesides, "A complete and effective move set for simplified protein folding," in *Proc. 7th Annu. Int. Conf. Research Comput. Molecular Biol.*, 2003, pp. 188–195.

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

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

[56] 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, Cybern.*, vol. 36, no. 3, pp. 534–545, Jun. 2006.

[57] H.-S. Kim and S.-B. Cho, "An efficient genetic algorithm with less fitness evaluation by clustering," in *Proc. CEC*, vol. 2. 2001, pp. 887–894.

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

[59] M. K. Islam and M. Chetty, "AI 2009: Advances in artificial intelligence," in *Novel Memetic Algorithm for Protein Structure Prediction* (Lecture Notes in Computer Science). Berlin/Heidelberg, Germany: Springer-Verlag, 2009, pp. 412–421.

[60] H.-P. Hsu, V. Mehra, W. Nadler, and P. Grassberger, "Growth algorithms for lattice heteropolymers at low temperatures," *J. Chem. Phys.*, vol. 118, no. 1, pp. 444–451, 2003.

[61] R. König and T. Dandekar, "Improving genetic algorithms for protein folding simulations by systematic crossover," *Biosystems*, vol. 50, no. 1, pp. 17–25, Apr. 1999.

[62] C. Cotta, "Protein structure prediction using evolutionary algorithms hybridized with backtracking," in *Artificial Neural Nets Problem Solving Methods* (Lecture Notes in Computer Science, vol. 2687), J. Mira and J. lvarez, Eds. Berlin/Heidelberg, Germany: Springer, 2003, pp. 1044–1044.

Md. Kamrul Islam (M'11) received the Ph.D. degree from the Gippsland School of Information Technology, Monash University, Churchill, Australia, in 2012.

Currently, he is an Adjunct Researcher with the Gippsland School of Information Technology and a Software Developer with Leap eLearning Pty, Ltd. His current research interests include artificial intelligence in bioinformatics, with particular interest in different behavioral patterns of evolutionary algorithms.

Madhu Chetty (SM'12) is currently the Deputy Head of the Gippsland School of Information Technology, Monash University, Churchill, Australia. He works in the areas of evolutionary computation, mathematical modeling of large-scale complex systems, and optimization. He has a specific interest in the application of computational intelligence techniques to problems from life sciences, i.e., protein structure prediction and systems biology. He and his team are currently focused on modeling G protein-coupled receptors and genetic networks for cyanobacteria.

Mr. Chetty has served as the Chair of the International Association for Pattern Recognition Technical Committee on Bioinformatics and the Vice Chair of the IEEE Computational Intelligence Society Technical Committee on Bioinformatics. He has been an Associate Editor of *Neurocomputing*.