An Enhanced Genetic Algorithm for *Ab Initio* Protein Structure Prediction

Mahmood A. Rashid, Firas Khatib, Md Tamjidul Hoque, and Abdul Sattar

***Abstract*—*In-vitro* methods for protein structure determination are time-consuming, cost-intensive, and failure-prone. Because of these expenses, alternative computer-based predictive methods have emerged. Predicting a protein's 3-D structure from only its amino acid sequence—also known as *ab initio* protein structure prediction (PSP)—is computationally demanding because the search space is astronomically large and energy models are extremely complex. Some successes have been achieved in predictive methods but these are limited to small sized proteins (around 100 amino acids); thus, developing efficient algorithms, reducing the search space, and designing effective search guidance heuristics are necessary to study large sized proteins. An on-lattice model can be a better ground for rapidly developing and measuring the performance of a new algorithm, and hence we consider this model for larger proteins (>150 amino acids) to enhance the genetic algorithms (GAs) framework. In this paper, we formulate PSP as a combinatorial optimization problem that uses 3-D face-centered-cubic lattice coordinates to reduce the search space and hydrophobic-polar energy model to guide the search. The whole optimization process is controlled by an enhanced GA framework with four enhanced features: 1) an exhaustive generation approach to diversify the search; 2) a novel hydrophobic core-directed macro-mutation operator to intensify the search; 3) a per-generation duplication elimination strategy to prevent early convergence; and 4) a random-walk technique to recover from stagnation. On a set of standard benchmark proteins, our algorithm significantly outperforms state-of-the-art algorithms. We also experimentally show that our algorithm is robust enough to produce very similar results regardless of different parameter settings.**

***Index Terms*—Combinatorial optimization, face-centered-cubic (FCC) lattice, genetic algorithms (GAs), HP model, macro-mutation, protein structure prediction (PSP), random-walk.**

Manuscript received May 20, 2015; revised August 30, 2015 and November 22, 2015; accepted November 27, 2015. Date of publication December 3, 2015; date of current version July 26, 2016. The work of M. A. Rashid and F. Khatib was supported by the Department of Computer and Information Science at the University of Massachusetts Dartmouth. The work of M. T. Hoque was supported by the Louisiana Board of Regents through the Board of Regents Support Fund under Grant LEQSF(2013-16)-RD-A-19. The work of M. A. Rashid and A. Sattar supported by the National ICT Australia and Institute for Integrated and Intelligent Systems at Griffith University, Australia.

M. A. Rashid is with the School of Computing, Information and Mathematical Sciences, University of the South Pacific, Laucala Bay, Suva, Fiji, and also with the Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, QLD 4111, Australia (e-mail: mahmood.rashid@usp.ac.fj).

F. Khatib is with the Department of Computer and Information Science, University of Massachusetts Dartmouth, Dartmouth, MA 02747 USA (e-mail: fkhatib@umassd.edu).

M. T. Hoque is with the Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA (e-mail: thoque@uno.edu).

A. Sattar is with the Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, QLD 4111, Australia (e-mail: a.sattar@griffith.edu.au).

Digital Object Identifier 10.1109/TEVC.2015.2505317

I. Introduction

PROTEINS are amongst the most important macromolecules in all living organisms. More than half of the dry weight of a cell is protein [1], [2]. Proteins are the sequential-chains of amino acids connected together by single peptide bonds. These connected chains fold into their functional 3-D structures [3] and regulate the cellular activities in living organisms to keep them alive. The function of a protein greatly depends on its folded 3-D structure (also known as native structure), which has the lowest possible free energy—the approximation of interaction energies amongst the amino acids in a protein [4]. There are, however, some exceptions such as proteins of prion domain that have multiple functional structures [5]. Many fatal diseases such as prion disease, Alzheimer's disease, Huntington's disease, Parkinson's disease, diabetes, and cancer are associated with the aggregation of nonfunctional proteins due to misfolding [6]–[9]. The 3-D structures of proteins are decidedly important in rational drug design [10], [11], protein engineering [12], [13], and biotechnology [14], [15]; thus, the protein structure prediction (PSP) has emerged as an important multidisciplinary research problem.

Among the *in-vitro* methods, X-ray crystallography, nuclear magnetic resonance spectroscopy, and electron microscopy are widely used for determining protein structures. X-ray crystallography, which is considered to be one of the available relatively accurate methods, requires the target proteins to undergo a complex crystallization process. As a result, X-ray crystallography often remains less feasible for proteins which are hard to crystallize such as membrane proteins [16]. In addition, these experimental methods are time consuming, cost-intensive, and failure-prone and are not sufficient to fill the gap between the number of known protein sequences ($\approx$50M[1]) and the number of solved structures ($\approx$0.1M[2]). Therefore, developing efficient algorithms for the predictive methods is thought to be a possible solution for bridging this gap. Again, predicting the 3-D structure of a protein from only its amino acid sequence is computationally demanding [17] because the search space is astronomically large and the confrontational-energy space is hugely complex. Some successes are achieved in high resolution predictive methods [18]–[22], but those are limited to mostly of protein-length <100 amino acids and some cases $\approx$150 amino acids [23], [24]. However, the average length of proteins could be more for some important domains

[1]The EBI: http://www.ebi.ac.uk/uniprot/TrEMBLstats/ (as of Jul. 22, 2015).
[2]The PDB: http://www.pdb.org/ (as of Jul. 22, 2015).

to study, such as 270 ± 9 for archaeal, 330 ± 5 for bacteria, and 449 ± 25 for eukaryotes [25], [26]. This limitation of high resolution models eventually motivates us to consider low resolution-based models for PSP. Because of the simplicity, a lattice-based model helps develop and validate new algorithms fast particularly for larger proteins (>150 amino acids). The high performing algorithms on lattice models could then be tested by implementing on high-resolution models. In a work, Higgs *et al.* [27] applied a similar strategy by replacing the Monte Carlo (MC) or conformational space annealing used in Rosetta with a genetic algorithm (GA) and found improved results.

According to Levinthal's paradox [28], it is impossible for a protein to go through all of its possible conformations to reach the correct native state, since it would take an astronomically large time, while in reality proteins take only a few seconds or less to reach their native state. This implies that there should be a folding pathway. In addition, Anfinsen's thermodynamic hypothesis [29] states that, at least for small globular proteins, the native structure is determined only by the protein's amino acid sequence. This also states that, at normal conditions (temperature, solvent concentration and composition, etc.), when folding occurs, the native structure is a unique, stable, and kinetically accessible minimum of the free energy.

Levinthal's paradox and Anfinsen's hypothesis are the bases of formulating *ab initio* PSP as an optimization problem. Given a protein's amino acid sequence, the problem is to find the 3-D structure of a protein such that the total interaction energy amongst the amino acids in the sequence is minimized. For this combinatorial optimization problem, we challenged ourselves to propose an enhanced GA to find a combination of points on the 3-D face-centered-cubic (FCC) lattice space to map the amino acids by minimizing the interaction energy.

The state-of-the-art results on the FCC lattice-based hydrophobic-polar (HP) energy model have been achieved by local search (LS) methods in [30]–[33]. On the other hand, GAs [34] and tabu search [35] found promising results on 2-D and 3-D hexagonal lattice-based HP models. In general, the success of either GA- or LS-based methods crucially depends on the balance of the diversification and the intensification of the sampling over the search space. Moreover, these algorithms often get stuck in local minima and perform poorly on large sized proteins. To progress further with these algorithms will require us to address the aforementioned issues appropriately, and an on-lattice low resolution model can provide such a feasible paradigm to design efficient sampling algorithms.

In this paper, we propose a population-based algorithm within the GA framework—named GAPlus based on our preliminary work [36]—for backbone-only PSP with the hope of studying larger proteins. We use an HP energy model and a 3-D FCC lattice to simplify the problem. In GAPlus, we introduce: 1) an exhaustive generation approach to diversify the search; 2) a novel hydrophobic core-directed macro-mutation operator to intensify the search; 3) a per-generation duplication elimination strategy to prevent early convergence; and 4) a random-walk technique to recover from stagnation. On a set of benchmark proteins, the GAPlus significantly outperforms state-of-the-art algorithms for PSP on the same models. In conjunction with the comparative results, we present a detailed empirical study on GA operators and GA parameters.

The rest of the paper is organized as follows. Section II reviews background knowledge, Section III discusses related work on PSP, Section IV describes our GAPlus framework, Section V presents the experimental results and analyses, Section VI presents a discussion on the enhanced features of GAPlus and their suitability of adaptation in the other classes of optimization problems, and finally, Section VII draws the conclusion and outlines our future research.

II. Background

The computer-based predictive methods are mostly available for template-based modeling (TBM) and *ab initio* modeling. In TBM, if a solved structure is found in the protein data bank (PDB) that has a significant sequential identity ($\geq$35%) with the target [37], [38], it is taken as a template. Then, different algorithms [39]–[42] are applied to refine the template to obtain the structure of the target protein.

However, *ab initio* modeling depends only on the amino acid sequence of the protein. The high resolution *ab initio* models [18], [43] consider all atoms of the backbone with their side chains, a complex energy model (such as molecular dynamics), and an internal or angular coordinates in 3-D space. On the other hand, the low resolution models [30]–[33] consider an amino acid with its side-chain as one residue of the backbone, a relatively simpler energy function (such as HP), and on-lattice or off-lattice representation. Our work is based on low resolution *ab initio* modeling that uses the HP energy model for evaluating the structures and the FCC lattice points on 3-D space for mapping the structures maintaining a self-avoiding-walk constraint within an enhanced GA framework.

A. Self-Avoiding Walk

In lattice-based protein representations, the amino acids of a given sequence are mapped on lattice points to build the protein backbone satisfying a self-avoiding-walk constraint. This constraint ensures no revisitation of any lattice point during the sequence mapping.

B. 3-D FCC Lattice

The FCC lattice has the highest packing density compared to the other existing lattices [44]. The hexagonal closed pack (HCP) lattice—known as cuboctahedron [34]—also has 12 neighbors corresponding to 12 basis vectors with real-numbered coordinates that causes the loss of structural precision during structure mapping. Therefore, we do not consider the HCP lattice in this paper.

1) Topological Neighbors of 3-D FCC Lattice Point: In FCC lattice space, each lattice point has 12 neighbors with corresponding 12 basis vectors as shown in Fig. 1. The following 12 basis vectors, which are the topological neighbors of the origin $(0, 0, 0)$, are the building blocks (genes) of our

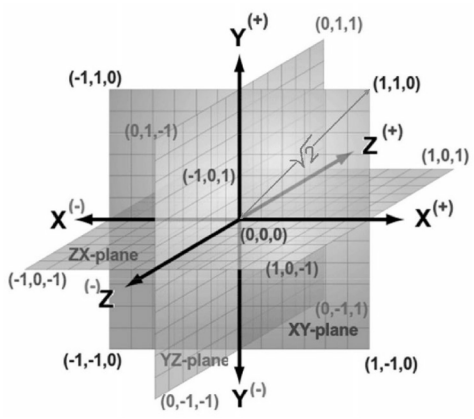
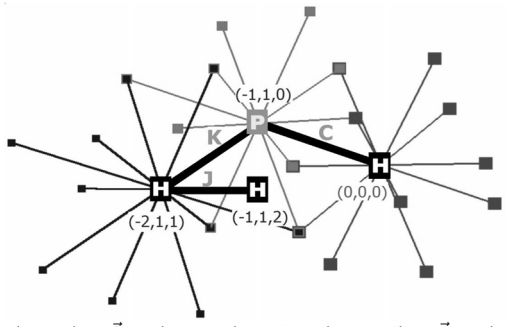


Fig. 1. 3-D FCC lattice space with 12 basis vectors on the Cartesian coordinates.



i) $(0,0,0)+\vec{C}\rightarrow(-1,1,0)$ *ii)* $(-1,1,0)+\vec{K}\rightarrow(-2,1,1)$
iii) $(-2,1,1)+\vec{J}\rightarrow(-1,1,2)$

Fig. 2. DOF in 3-D FCC lattice space. The backbone mapping of a 4-residue (HPHH) sequence segment is shown by the thicker lines. $\vec{C}$, $\vec{K}$, and $\vec{J}$ are the basis vectors as stated in Section II-B. The encoded string for the segment is *CKJ*. The encoding and decoding procedures are discussed in Section IV-B2.

encoded solutions (chromosomes or individuals):

$A=(1,1,0)$ $B=(-1,-1,0)$ $C=(-1,1,0)$ $D=(1,-1,0)$
$E=(0,1,1)$ $F=(0,-1,-1)$ $G=(0,1,-1)$ $H=(0,-1,1)$
$I=(-1,0,-1)$ $J=(1,0,1)$ $K=(-1,0,1)$ $L=(1,0,-1)$.

We map the conformations on the 3-D FCC lattice points using a sequence of basis vectors (see Fig. 2).

2) Degree of Freedom and Computational Complexity: The local degree of freedom (DOF) on 3-D FCC lattice points varies based on the amino acid positions in the sequence. Starting from a random point (any of the 12 FCC lattice points), the DOF of the first and the second amino acid in the series is 12. However, the DOF for all other amino acids starting from 3 onward is 11 (see Fig. 2). For sampling all possible conformations of a sequence with N amino acids, an algorithm needs to explore a search space that consists of $12^2\times 11^{(N-2)}\approx 11^N$ valid combinations.

C. HP Energy Model

The 20 primary constituent amino acids of proteins are broadly divided into two categories, based on the hydrophobicity: 1) the hydrophobic amino acids are denoted as H (Gly, Ala, Pro, Val, Leu, Ile, Met, Phe, Tyr, Trp) and 2) the hydrophilic or polar amino acids are denoted as P (Ser, Thr, Cys, Asn, Gln, Lys, His, Arg, Asp, Glu). In the HP model [45], when two

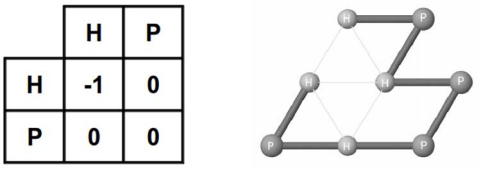


Fig. 3. (a) HP energy model [45]. (b) H–H contacts in HP model for a random sequence HPHPPHPH on a 2-D triangular lattice for convenience of depiction.

Algorithm 1: Genetic Algorithm

1 initialize population and evaluate individuals;
2 **while** *(!stopCondition)* **do**
3 select the best-fit individuals for reproduction;
4 breed new individuals through crossover and mutation operations;
5 evaluate new generated individuals;
6 replace least-fit individuals with the new better individuals;

nonconsecutive hydrophobic amino acids become topological neighbors, they contribute a certain amount of negative energy, which, for simplicity, is shown as -1 in Fig. 3(a). The total energy (E) of a conformation based on the HP model is the sum of the contributions from all pairs of nonconsecutive hydrophobic amino acids as shown in (1). In Fig. 3(b), the number of such H–H contacts is 5. Therefore, the fitness of the structure of the sequence HPHPPHPH is -5 in the HP energy model

$$E=\sum_{i<j-1}c_{ij}\times e_{ij}. \tag{1}$$

Here, $c_{ij}=1$ if amino acids i and j are nonconsecutive neighbors on the lattice, otherwise 0; and $e_{ij}=-1$ if the ith and jth amino acids are hydrophobic, otherwise 0. Note that (1) is used as the minimization function for our GA-based combinatorial optimization.

D. Genetic Algorithms

A GA maintains a set of solutions known as population (line 1 in Algorithm 1). In each generation, the GA generates a new population from the current population using a given set of genetic operators known as crossover and mutation (line 4 in Algorithm 1). The algorithm then replaces the inferior solutions with the newly generated superior solutions to get a better current population (line 6 in Algorithm 1). A typical crossover operator splits two solutions at a randomly selected crossover point and exchanges parts between them [Fig. 4(a)]. Conversely, a typical mutation operator alters a solution at a random point [Fig. 4(b)]. Here the conformations are regarded as the individuals (chromosomes) of the GA population.

III. Related Work

In the brief history of *ab initio* predictive methods, some successes have been achieved for high resolution-based real

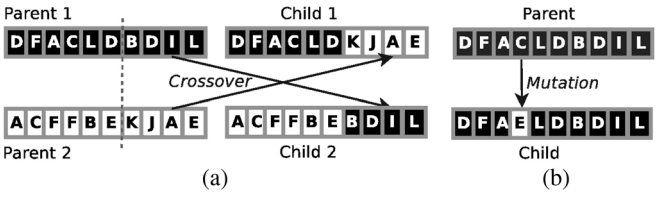


Fig. 4. Typical (a) crossover and (b) mutation operators.

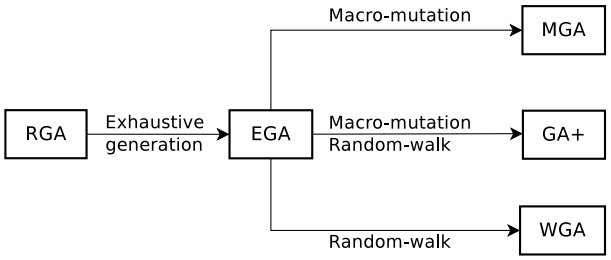


Fig. 5. Relations amongst the GA variants. The RGA is the typical implementation of GA with random generation approach; the EGA inherits all of the features of the RGA with an additional exhaustive generation approach; the MGA inherits all of the features of the EGA with an additional macro-mutation operator; the WGA inherits all of the features of the EGA with an additional random-walk-based stagnation recovery technique; and the GAPlus inherits all of the features of the EGA with the additional macro-mutation operator and random-walk-based stagnation recovery technique.

protein modeling [18]–[21]. However, these successes are limited to the small sized proteins. Our low-resolution model is designed to target large sized proteins and therefore, we did not compare our outcomes with the high resolution models at this stage.

Different types of meta-heuristics have been used in solving the PSP problem. These include MC simulation [46], simulated annealing [47], GAs [48], [49], tabu search with GA [35], tabu search with hill climbing [50], ant colony optimization [51], particle swarm optimization [52], [53], immune algorithms [54], tabu-based stochastic LS [30], [32], and constraint programming [31], [55].

Cebrián *et al.* [30] used tabu-based LS, and Shatabda *et al.* [32] used memory-based LS with tabu-based heuristics and achieved state-of-the-art results. However, Dotu *et al.* [31] used constraint programming and found promising results. To develop the constraint-based PSP (CPSP) tools, Mann *et al.* [55] applied an exact and complete algorithm for conformational search. The CPSP tools can find the optimal solution if the target protein has a matching hydrophobic-core stored in the CPSP database.

Amongst the population-based approaches, Unger and Moult [48] applied GAs to PSP and found their method to be more promising than MC-based methods [46]. GAs have been used by Hoque *et al.* [34] for cubic and 3-D HCP lattices. A twin-removal operator was introduced in [56] to remove duplicates from the GA population. The few other simplified methods for PSP are also available [57]–[60] but they use different energy models [61], [62] and thus, we keep these methods out of our comparisons.

In 3-D FCC lattice and HP energy-based PSP, the recent state-of-the-art results have been achieved by tabu-guided LS algorithms [30], [32], [33] and by memetic algorithms [63]. In order to confirm the efficacy of our proposed algorithm, we have compared our results with the results of these state-of-the-art approaches.

IV. Our Approach

In this section, we present an enhanced GA (GAPlus) with detailed implementation of: 1) an exhaustive generation approach to diversify the search; 2) a novel hydrophobic core-directed macro-mutation operator to intensify the search; 3) a per-generation duplication elimination strategy to prevent early convergence; and 4) a random-walk technique to recover from stagnation.

A. GA Variants

The relations amongst the GA variants are shown in Fig. 5. We have implemented five variants of GA for PSP to demonstrate the effectiveness of the enhanced features as listed below.

1) RGA (R denotes randomness) is a typical implementation of a GA. It uses crossover and mutation operators. The crossover operator includes only single-point crossovers. The mutation operators include rotation, diagonal moves, pull moves, and tilt moves. The operator selection process is random. The crossover and mutation points are also selected randomly. RGA has a duplicate elimination policy (see Section IV-B5). Further details are in Section IV-B.
2) EGA (E denotes exhaustiveness) inherits all of the features of the RGA. Additionally, a novel exhaustive generation approach is used. In EGA, the random selection of crossover and mutation points is replaced with an exhaustive selection approach as described in Algorithms 9 and 10. The EGA is the base of WGA, MGA, and GAPlus. This variant demonstrates the effectiveness of our exhaustive generation approach. Further details are in Section IV-C2.
3) MGA (M denotes macro-mutation) inherits all of the features of the EGA. Additionally, a novel macro-mutation operator is applied. The macro-mutation operator is used as a new mutation operator like other mutation operators [see Fig. 6(b)–(e)]. The macro-mutation operator helps form hydrophobic-cores that hide the hydrophobic amino acids from water and expose the polar amino acids to the surface to be in contact with the surrounding water molecules [64]. This variant demonstrates the effectiveness of our hydrophobic-core directed macro-mutation operator. Further details are in Section IV-C3.
4) WGA (W denotes random-walk) inherits all of the features of the EGA. Additionally, at the point of stagnation, a novel random-walk-based stagnation recovery technique is applied. This variant demonstrates the effectiveness of our random-walk-based stagnation recovery approach. Further details are in Section IV-C4.
5) GAPlus inherits all of the features of EGA. Additionally, it includes the hydrophobic-core directed macro-mutation operator and the random-walk-based stagnation

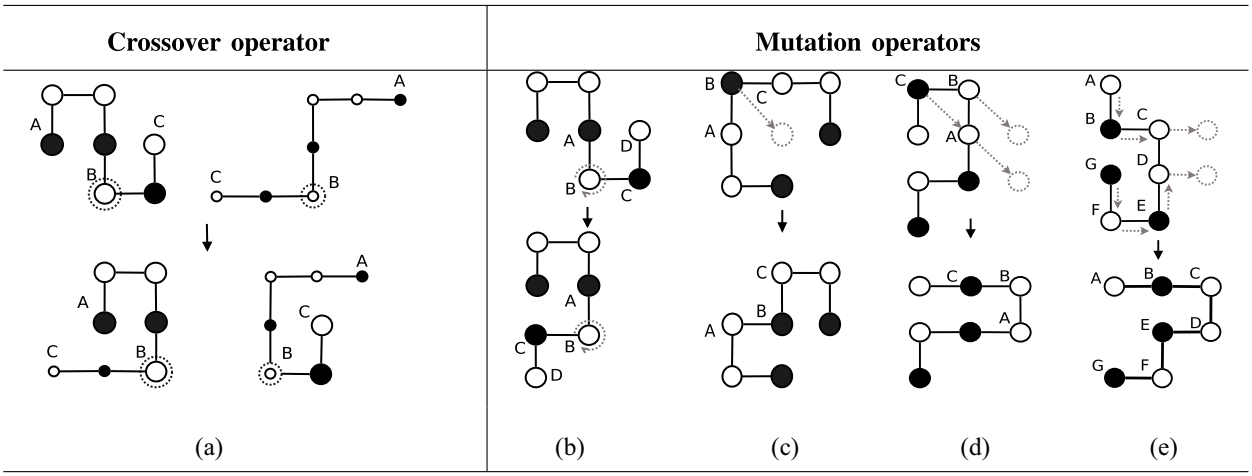


Fig. 6. Operators that are used in our GAPlus on 3-D FCC lattice space. For simplification, the figures are presented in 2-D space. The solid circles represent the hydrophobic amino acids and the hollow circles represent the hydrophilic amino acids. (a) Crossover. (b) Rotation. (c) Diagonal. (d) Pull. (e) Tilt.

recovery technique. The inclusion of these enhanced features helps our GAPlus produce state-of-the-art results (see Section V-A) for lattice-based PSP. The further details are presented in Section IV-C6.

The executable binaries of the GA variants are available online at http://cs.uno.edu/~tamjid/Software.html, under the heading: GAPlus.

B. RGA Framework

A typical randomized GA framework is implemented and denoted as the RGA. The pseudocode of RGA is presented in Algorithm 2. The RGA starts by generating an initial population (line 6). The operator selection process is random in the RGA (line 8). For mutation operators, each individual undergoes the offspring generation process (line 10). For the crossover operator, the two parents are selected randomly (line 16). The mutation (line 11) and crossover (line 17) points are also selected randomly. The randomized mutation or crossover operators are applied at line 12 or line 18 respectively. The duplicate elimination process is performed in line 13 or line 19. No explicit stagnation recovery technique is implemented in the RGA. The implementation details of the RGA are presented below.

Algorithm 2: RGA()

```
1  //op: Operators; c, c': Conformations
2  //opR: Operator selection rate
3  //curP, newP: Current & new population
4  //N: No. of amino acids in the sequence
5  //pos: Mutation or crossover point
6  curP ←initialize(repeat);
7  foreach (generation until timeout) do
8      selectOperator(op, opR);
9      if (mutation(op)) then
10         foreach (c∈ curP) do
11             pos ←randMutationPoint(0, N − 1);
12             c' ←mutate(c, pos);
13             add(newP, c')
14     else
15         while (¬ full(newP)) do
16             c1, c2 ← randConfs(curP);
17             pos ←randCrossoverPoint(1, N − 2);
18             c'1, c'2 ← crossOver(c1, c2, pos);
19             add(newP,c'1, c'2)
20     curP ← newP
21 return bestConformation(curP)
```

1) Implementation of the Primitive Operators: In our GA, we implemented one primitive crossover operator and four primitive mutation operators (see Fig. 6). The mutation operators are rotation, diagonal moves, pull moves, and tilt moves. The primitive operators are described below.

a) Crossover operators: The crossover operators are applied on two selected parent conformations to exchange their parts to generate child conformations. At a given crossover point [dotted circle in Fig. 6(a)], two parent conformations exchange their parts and generate on the two child conformations. The success rate of the crossover operator decreases with increasing the compactness of the structure.

b) Mutation operators: The mutation operators are applied on a single conformation. The alteration can occur either on a single amino acid or on a series of amino acids of the conformation. The primitive mutation operators [as shown in Fig. 6(b)–(e)] are described below.

1) *Rotation:* One part of a given conformation is rotated around a selected point [Fig. 6(b)]. This move is mostly effective at the beginning of the search.
2) *Diagonal Move:* Given three consecutive amino acids at lattice points A, B, and C, a diagonal move at position B takes the corresponding amino acid diagonally to a free position [Fig. 6(c)]. Diagonal moves are very effective on FCC lattice [30], [31] points.
3) *Pull Moves:* The amino acids at points A and B are pulled to the free points [Fig. 6(d)] and the connected amino acids are pulled as well to get a valid conformation. Pull moves [65] are local, complete, and reversible.

Algorithm 3: encode(c)

```
1  //c: Conformation;AA: Amino acid array
2  //S: Encoded string;s: Vector symbol
3  //N: No. of amino acids in the sequence
4  S ← {}//the length of S is (N − 1);
5  AA ←getAminoAcid(c);
6  for (i ← 1 to N − 1) do
7  |   vec ←getAbsoluteVector(AA[i], AA[i − 1]) ;
8  |   s ←getSymbol(vec) ;
9  |   S ←concat(S, s);
10 return S
```

Algorithm 4: decode(S)

```
1  //c: Conformation;AA: Amino acid array
2  //S: Encoded string;s: Vector symbol
3  //N: No. of amino acids in the sequence
4  N ← Length(S) + 1;
5  AA[0] ← (0, 0, 0);
6  for (i ← 1 to N − 1) do
7  |   s ←getSymbol(S, i);
8  |   vec ←getAbsoluteVector(s);
9  |   point ←getNextPoint(point,vec);
10 |   AA[i] ← point;
11 c ←buildConformation(AA);
12 evaluate(c);
13 return c ;
```

Algorithm 5: initialize(*repeat*)

```
1  //c: Conformation;AA: Amino acid array
2  //initP: Initial population
3  //N: No. of amino acids in the sequence
4  while (¬ full(newP)) do
5  |   AA[0] ← AminoAcid(0,0,0);
6  |   while (try ≤ repeat) do
7  |   |   for (i ← 1 to N − 1) do
8  |   |   |   k ← getRandom(1,12);
9  |   |   |   point ← AA[i-1] + basisVec[k − 1];
10 |   |   |   if (¬ free(point)) then break;
11 |   |   |   AA[i] ← AminoAcid(point);
12 |   if (mapping succeeded) then
13 |   |   c ←buildConformation(AA);
14 |   else
15 |   |   c ←deterministicConformation();
16 |   encode(c);
17 |   evaluate(c);
18 |   add(initP, c);
19 return initP;
```

Pull moves are very effective when the conformation becomes compact.

4) *Tilt Moves:* Two or more consecutive amino acids connected in a straight line are moved by a tilt move to immediately parallel lattice positions [49]. Tilt-moves pull the conformation from both sides until a valid conformation is found. In Fig. 6(e), the amino acids at points *C* and *D* are moved and, subsequently, other amino acids from both sides are moved as well.

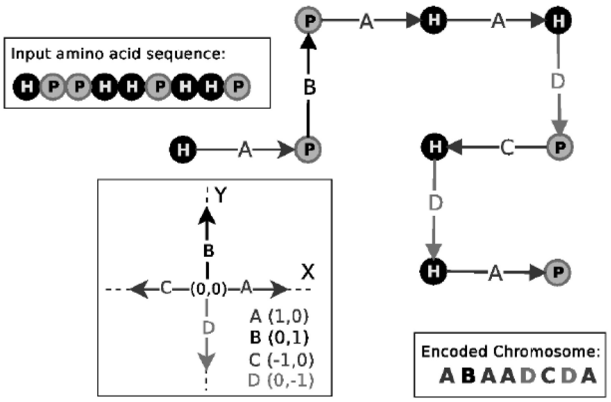
2) Conformation Encoding and Decoding: We represent conformations with 3-D coordinates and relative encodings. Each time, when a valid conformation is generated by applying genetic operators, we encode the conformation in a string of 12 different characters [A–L] representing the 12 absolute directions on a 3-D FCC lattice space (see Section II-B). For each valid conformation, we store both the coordinates of the amino acids and the encoded string. While the coordinates help us determine whether a point on the lattice is free or not, the relative encodings help validate conformations and identify and eliminate duplicate ones.

a) Conformation encoding: In our implementation, we denote encoding as the procedure of generating a string to present a conformation using character notations, where each notation is the position of an amino acid in a 3-D space. The first character of the encoded string will be the absolute direction of the second amino acid from the first amino acid. The second character of the encoded string will be the absolute direction of the third amino acid from the second amino acid, and so on. The length of the encoded string will be one less than the number of amino acids in the sequence (i.e., $N-1$). Fig. 7 presents a simplified encoding procedure in 2-D space. Algorithm 3 presents the encoding procedure.

b) Conformation decoding: In our implementation, we denote decoding as the procedure of extracting the coordinates of each amino acid from the encoded string. Considering (0, 0, 0) as the coordinate of the first amino acid of the sequence, the first character of the encoded string will provide the absolute direction of the second amino acid and hence the coordinate of the second amino acid is determined. The second character of the encoded string will provide the absolute direction of the third amino acid and hence the coordinate of the third amino acid is determined, and so on. Algorithm 4 presents the decoding procedures.

3) Initializing the RGA Population: The RGA starts with an initial population, which is a set of feasible conformations. We generate initial conformations following a self-avoiding walk on the FCC lattice points. The pseudocode of the initialization algorithm is presented in Algorithm 5. It places the first amino acid at (0, 0, 0) (line 5) and randomly selects a basis vector to place the next amino acid at a neighboring free lattice point (lines 8–11). The amino acid position mapping continues until a self-avoiding walk is found for the whole protein sequence within a given number of iterations. The conformation is encoded (line 16), evaluated (line 17), and added into the initial population list (line 18) if it does not exist in the list. If no valid conformation is found within the given number of iterations, a deterministic structure—an amino acid chain with unidirectional growth (e.g., *AAAA*... as the encoded chromosome)—is built (line 15).

4) Evaluating the Solutions: For each iteration, the conformation is evaluated by calculating the number of contacts (topological neighbors), where the two amino acids are nonconsecutive and are both hydrophobic. The pseudocode in



Pos	*Curr*		*Vec*		*Next*
1		Start			H1(0, 0)
2	H1(0, 0)	+	A(1, 0)	→	P2(1, 0)
3	P2(1, 0)	+	B(0, 1)	→	P3(1, 1)
4	P3(1, 1)	+	A(1, 0)	→	H4(2, 1)
5	H4(2, 1)	+	A(1, 0)	→	H5(3, 1)
6	H5(3, 1)	+	D(0, −1)	→	P6(3, 0)
7	P6(3, 0)	+	C(−1, 0)	→	H7(2, 0)
8	H7(2, 0)	+	D(0, −1)	→	H8(2, −1)
9	H8(2, −1)	+	A(1, 0)	→	P9(3, −1)

Fig. 7. Chromosome encoding procedures. For simplification, the figure is presented in 2-D space with four possible vectors A-right, B-top, C-left, and D-bottom. In the chart, the Pos, Curr, Vec, and Next represent the amino acid positions, current amino acids, basis vectors, and next amino acids, respectively. The next position [e.g., P2(1, 0)] is obtained adding the next direction vector [e.g., A(1, 0)] with the coordinates of the current amino acid [e.g., H1(0, 0)].

Algorithm 6: evaluate(c)

```
1  //c: Conformation;AA: Amino acid array
2  //N: No. of amino acids in the sequence
3  AA ←getAminoAcid(c);
4  fitness ← 0;
5  for (i ← 0 to N − 1) do
6  |  for (j ← i + 2 to N − 1) do
7  |  |  if (AcidType[i] = AcidType[i] =′ H′) then
8  |  |  |  pointI ← AA[i];
9  |  |  |  pointJ ← AA[j];
10 |  |  |  sqrD ← getSqrDist(pointI, pointJ);
11 |  |  |  if (sqrD = 2) then
12 |  |  |  |  fitness ← fitness − 1;
13 update(c, fitness);
14 return c;
```

Algorithm 7: add(pop, c)

```
1  //pop: population, c, c′: Conformations
2  exist ←false;
3  foreach (c′ ∈ pop) do
4  |  if (c′ = c) then
5  |  |  exist ←true;
6  |  |  break;
7  if (¬exist) then pop ←add(conf)
```

Algorithm 6 presents the procedure for calculating the contact energy of a given conformation. The contact potentials are found in the HP energy model as shown in Fig. 3. The algorithm calculates the contacts based on (1). The Euclidean distance between two topological neighbors on the FCC lattice space is $\sqrt{2}$. Therefore, all H amino acids which are nonconsecutive and are $\sqrt{2}$ lattice distance apart are counted (line 12) to get the fitness of the conformation.

5) Removing the Duplicate Solutions: In our implementation, no duplicate solution is allowed in the population in any instant of the search. Every time the RGA generates a solution, it is compared with the new population to check if it already exists. The new solution is included in the list if and only if the solution does not exist in the new population. In Algorithm 2, line 13 or line 19 prevents the identical—100% similar—solutions from being added to the new population by using Algorithm 7. This is how the duplicate-free new population becomes the current population in the next generation of the RGA (line 20).

6) Selecting the RGA Operators: The probability distribution to select operators is chosen intuitively. The crossover and mutation operators are selected with the probabilities of 20% and 80%, respectively. The four implemented primitive mutation operators are rotation, diagonal-move, pull-move, and tilt-move. For a particular generation, only one mutation operator is selected randomly from the four with a probability of 80%. At this stage of experimentation, we do not consider the optimality of operator selection probabilities. We present operator profiling in detail in Section V.

7) Choosing the RGA Population Size: The population size is chosen intuitively. For the time being, we use 100, 80, 60, and 50 as the population sizes for the protein sequences having ≤ 50, ≤ 100, ≤ 200, and ≤ 300 amino acids, respectively. At this stage of experimentation we do not consider the optimality of GA population size. We present a rigorous study on GA population size in Section V.

8) Housekeeping Between Generations: The globally best (i.e., top elite) solution is always preserved. At the end of building a new generation, before shifting the new population to the current population, the best solution of the current population is compared with the globally best solution in terms of the free energy level. If the free energy of the current best is better than that of the globally best, then the globally best solution is replaced by the current best solution.

C. GAPlus Framework

We make three major enhancements on typical GAs: 1) an exhaustive generation to diversify population; 2) a macromutation operator to intensify the population; and 3) a random-walk for stagnation recovery. We call the enhanced GA, GAPlus. The pseudocode of GAPlus is shown in Algorithm 8.

Algorithm 8: GAPlus()

```
1  //op: Operators, c, c': Conformations
2  //opR: Operator selection probabilities
3  //curP, newP: Current & new populations
4  //rwT: Non-improving generation counter
5  curP ←initialize(repeat);
6  foreach (generation until timeout) do
7      selectOperator(op, opR);
8      if (mutation(op)) then
9          foreach (c ∈ curP) do
10             if (¬ isMacro) then
11                 c' ←mutConf(c);
12             else
13                 c' ←macroMutation(c, repeat) ;
14             add(newP, c')
15     else
16         while (¬ full(newP)) do
17             c1, c2 ← randomConfs(curP);
18             c'1, c'2 ←crossConf(c1, c2);
19             add(newP,c'1, c'2)
20     if (¬ improved(newP,rwT)) then
21         curP ←randomWalk(newP)
22     else
23         curP ← newP
24 return bestConformation(curP)
```

Algorithm 9: mutConf(c)

```
1  // c: Conformation; pos: Mutation point
2  addToList(list, c);
3  for (pos ← 0 to N − 1) do
4      c' ←applyOperator(c, pos);
5      addToList(list, c');
6  return bestConformation(list)
```

Algorithm 10: crossConfs(c_1, c_2)

```
1  // c1, c2, c'1, c'2: Conformations; pos: Crossover point
2  addToList(list, c1, c2);
3  for (pos ← 1 to N − 2) do
4      c'1, c'2 ←applyOperator(c1, c2, pos);
5      addToList(list, c1, c2, c'1, c'2);
6  return bestTwoConformations(list)
```

Note that our GAPlus is different from a typical GA (e.g., RGA) in a number of ways. The distinguishable difference between RGA and GAPlus are listed below.

1) At lines 11 and 18, the GAPlus applies crossover or mutation operators exhaustively at each amino acid position (line 3 in Algorithm 9, line 3 in Algorithm 10). This exhaustiveness is the core feature of EGA.
2) Besides primitive GA operators, at line 13, the GAPlus introduces a macro-mutation operator (Algorithm 11) in a similar fashion of the other mutation operators. This macro-mutation is the core feature of MGA.
3) Besides duplicate elimination, at line 21, the GAPlus uses a random-walk algorithm (Algorithm 12) to recover from stagnation. This random-walk is the core feature of WGA.

1) Different Selection Strategies in GAPlus: All of the following processes in the GAPlus framework involve their purpose-specific selection strategies.

a) Population size selection: For the time being, we choose the population size as used in RGA (see Section IV-B7). We present a detailed analysis on the GA population size in Section V.

b) Operator selection: In our implementation, we intuitively assign a 20% selection probability for one crossover operator and 80% for five mutation operators (including the macro-mutation) with equal selection probabilities (20% of the remaining 80% for each mutation operator). Only the selected operator is applied to build the population for the next generation.

c) Parent selection: If the selected operator is mutation then every individual in the population has been passed through the mutation operator to build the population for the next generation. However, for the crossover operator, two parents are selected randomly until all individuals in the population have been passed through the crossover operator to build the population for the next generation.

d) Point of operation selection: The point of operation is chosen exhaustively starting from the first to the last amino acid of the sequence for the mutation operator and from the second to the second-last amino acid of the sequence for the crossover operator until finding the acceptable children.

e) Next generation selection: The fitness is the measure of selecting a solution for the next generation. When mutation is chosen to build the next generation, the parent and all the children resulting from the exhaustive mutation are added to a list. The fittest solution from the list then replaces the parent. Similarly, when crossover is chosen, the two parents and all the children resulting from the exhaustive crossover are added to a list. The two best fit solutions from the list then replace the two parents.

f) Random-walk timing selection: We evaluate and compare the current best solution with the so-far global-best solution at the end of each generation. When the global-best solution remains the same for a certain number of generations (a random value between 5 and 10 inclusive is considered), a random-walk is triggered.

2) Exhaustive Generation-Based Approach: To reduce the randomness, we introduce an exhaustive generation approach in the parent selection as well as in the mutation and crossover point selection.

a) Exhaustive mutation: For mutation operators, our algorithm adds one resultant conformation to the new population for each of the conformations in the current population. In Algorithm 9, the child conformations are generated by applying the genetic operator at each of the positions of the parent conformation (line 4). The resultant conformation of a mutation operation is either the parent conformation itself or a child depending on the quality (fitness and structure) of the conformations.

TABLE I
ENERGY VALUES OBTAINED BY RUNNING RGA AND EGA TO DEMONSTRATE THE EFFECTIVENESS OF THE EXHAUSTIVE GENERATION. COLUMN LB-FE PRESENTS THE LOWER BOUND OF FREE ENERGIES

Protein details				RGA			EGA			Time
Seq	Size	Hs	LB-FE	Best	Mean	STD	Best	Mean	STD	mins
H1	48	24	-69	-65	-58	2.63	⋆	**-67**	1.05	30
H2	48	24	-69	-63	-57	3.16	⋆	**-67**	0.93	30
H3	48	24	-72	-64	-57	3.49	**-71**	**-69**	1.29	30
H4	48	24	-71	-66	-58	3.55	⋆	**-69**	1.31	30
H5	48	24	-70	-65	-58	3.85	⋆	**-68**	1.10	30
F90_1	90	50	-168	-133	-120	5.73	**-159**	**-144**	6.54	120
F90_2	90	50	-168	-132	-124	6.09	**-155**	**-142**	5.84	120
F90_3	90	50	-167	-138	-124	8.00	**-158**	**-146**	5.78	120
F90_4	90	50	-168	-141	-124	7.01	**-158**	**-146**	5.03	120
F90_5	90	50	-167	-131	-120	7.02	**-158**	**-144**	6.43	120

⋆ denotes the lower bound of free energy is found.

b) Exhaustive crossover: For crossover operators, two resultant conformations are added to the new population for the two randomly selected parents from the current population. Crossover operators (Algorithm 10) generate child conformations by exhaustively selecting the crossover points on the parent conformations. In the GAPlus, a number of child conformations are generated and listed (line 5). The two best conformations in the list then become the resultant conformations (line 6 in Algorithm 10).

c) Effectiveness of exhaustive generation: To show the effectiveness of exhaustive generation experimentally, we tested our approach on ten small and medium sized benchmark proteins taken from the literature. In Table I, we present the results obtained from two variants of GA: 1) RGA that uses the random selection approach and 2) EGA that uses the exhaustive generation approach. From the experimental results, it is clear that the EGA significantly outperforms the RGA and, in some cases, the EGA is able to reach the lower bound of the free energy as denoted by "⋆" in column best, under the header EGA. In Table I, we present the global minimum energies in column best, the average of minimum energies in column Mean, and the standard deviation of the minimum energies in column STD. These values are obtained from 50 different trials in identical settings for all proteins. For both cases, the EGA performs better than RGA. This consistent performance demonstrates the effectiveness of the exhaustive generation.

3) Hydrophobic-Core Directed Macro-Mutation: In this section, we present a macro-mutation operator guided by the hydrophobic property of amino acids. The macro-mutation operator compresses the conformation and quickly forms the hydrophobic-core. In GAPlus, the macro-mutation operator is applied in a similar manner of the primitive mutation operators (line 13 in Algorithm 8).

a) Hydrophobic-core: Protein structures have hydrophobic-cores that hide the hydrophobic amino acids from water and expose the polar amino acids to the surface to be in contact with the surrounding water molecules [64]. The hydrophobic-core formation is an important objective of HP energy model-based PSP.

b) Hydrophobic-core center: The hydrophobic-core center is a virtual point in a 3-D space which is the centroid of the core formed by the hydrophobic amino acids in a given protein sequence. In the macro-mutation (Algorithm 11),

Algorithm 11: macroMutation(c, repeat)

```
1  // c: Conformation
2  hAA ←listHydrophobicAminoAcid(c);
3  pAA ←listPolarAminoAcid(c);
4  for (i ← 1to repeat) do
5      if (bernoulli(p)) then
6          T ← P
7      else
8          T ← H
9      if (T = P) then
10         foreach (aa ∈ pAA) do
11             point ←findFreePoint(aa);
12             if (¬empty(point)) then
13                 applyDiagonalMove(aa, point);
14                 updateConf(c, aa, point);
15                 break;
16     else
17         hcc ← findHCC();
18         foreach (aa ∈ hAA) do
19             d_old ←getDistance(aa, hcc);
20             point ←findFreePoint(aa);
21             if (¬empty(point)) then
22                 d_new ←getDistance(point, hcc);
23                 if (d_new ≤ d_old) then
24                     applyDiagonalMove(aa, point);
25                     updateConf(c, aa, point);
26                     break;
27 encode(c);
28 evaluate(c);
29 return c;
```

the hydrophobic core center (HCC) is calculated by finding the arithmetic means of x, y, and z coordinates of all hydrophobic amino acids as shown in the following equation:

$$x_{\text{hcc}} = \frac{1}{n_{\text{h}}}\sum_{i=1}^{n_{\text{h}}} x_i,\ \ y_{\text{hcc}} = \frac{1}{n_{\text{h}}}\sum_{i=1}^{n_{\text{h}}} y_i,\ \ z_{\text{hcc}} = \frac{1}{n_{\text{h}}}\sum_{i=1}^{n_{\text{h}}} z_i \tag{2}$$

where n_{h} is the number of H amino acids in the protein.

The HCC is used to calculate the Euclidean distance of all H amino acids using

$$d_i = \sqrt{(x_i - x_{\text{hcc}})^2 + (y_i - y_{\text{hcc}})^2 + (z_i - z_{\text{hcc}})^2} \tag{3}$$

where d_i is the Euclidean distance of the ith amino acid at (x_i, y_i, z_i) from the HCC at $(x_{\text{hcc}}, y_{\text{hcc}}, z_{\text{hcc}})$.

c) Macro-mutation operator: The macro-mutation operator is a composite operator that performs a series of diagonal moves on a given conformation to build the hydrophobic-core around the HCC as shown in Fig. 8. The pseudocode of the macro-mutation operator is shown in Algorithm 11.

The diagonal moves are applied repeatedly either at each P- or H-type amino acid position. Whether to apply the diagonal move on P- or H-type amino acids is determined by using a Bernoulli distribution (line 5) with probability p (intuitively we use $p = 20\%$ for P-type amino acids). For a P-type amino acid, the first successful diagonal move is

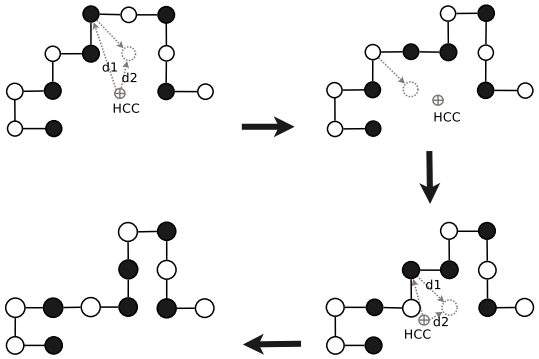


Fig. 8. Macro-mutation operator comprising a series of diagonal moves. For simplification, the figures are presented in 2-D space.

TABLE II
ENERGY VALUES OBTAINED BY RUNNING EGA AND MGA TO DEMONSTRATE THE EFFECTIVENESS OF THE MACRO-MUTATION OPERATOR. COLUMN LB-FE PRESENTS THE LOWER BOUND OF FREE ENERGIES

Protein details				EGA			MGA			Time
Seq	Size	Hs	LB-FE	Best	Mean	STD	Best	Mean	STD	mins
H1	48	24	-69	⋆	-67	1.05	⋆	**-67**	0.85	30
H2	48	24	-69	⋆	-67	0.93	⋆	**-68**	0.83	30
H3	48	24	-72	-71	-69	1.29	⋆	**-71**	1.06	30
H4	48	24	-71	⋆	-69	1.31	⋆	**-70**	1.06	30
H5	48	24	-70	⋆	-68	1.10	⋆	**-69**	0.75	30
F90_1	90	50	-168	-159	-144	6.54	**-165**	**-159**	2.69	120
F90_2	90	50	-168	-155	-142	5.84	**-164**	**-158**	2.82	120
F90_3	90	50	-167	-158	-146	5.78	**-163**	**-158**	2.69	120
F90_4	90	50	-168	-158	-146	5.03	**-165**	**-160**	2.30	120
F90_5	90	50	-167	-158	-144	6.43	**-166**	**-160**	2.47	120

⋆ denotes the lower bound of free energy is found.

considered (line 13). However, for an H-type amino acid, the first successful diagonal move (line 24) that does not increase the Euclidean distance (line 23) of the amino acid from the HCC, is taken. All the amino acids are traversed and the successful moves are considered as a single composite move. In this way, the macro-mutation operator compresses the conformation and quickly forms the hydrophobic-core by repeating the procedure (line 4).

d) Effectiveness of macro-mutation: In Table II, we present the results obtained from two variants of GA: 1) EGA that uses an exhaustive generation approach only and 2) MGA that uses all features of EGA along with the macro-mutation operator. From the experimental results, it is clear that the MGA significantly outperforms the EGA. In Table II, we present the global minimum energy (column Best), the average of minimum energies (column Mean), and the standard deviation of the minimum energies (column STD) obtained from 50 different runs of identical settings for all proteins using these two GA variants. For both cases, the MGA performs better than the EGA. This consistent performance demonstrates the effectiveness of the macro-mutation operator in GA.

4) Random-Walk-Based Stagnation Recovery: Search-based optimization algorithms often suffer from stagnancy [66]–[68]. For population-based algorithms, it occurs when the algorithms fail to build new individuals [69]. While exploring a search landscape, stagnation occurs when a search algorithm gets stuck or stalls in a local minimum, gets trapped in a valley, or gets lost in a plateau. This happens often in solving hard optimization problems such as PSP, particularly for large sized proteins. Therefore, developing an effective stagnation recovery technique is crucial for the conformational search.

Algorithm 12: randomWalk(*pop*)

```
1 // pop: Population; c: Conformation
2 foreach (c ∈ pop) do
3 |   isFound ← false;
4 |   while (¬isFound) do
5 |   |   for (pos ← 0 to N − 1) do
6 |   |   |   applyPullMove(c, pos);
7 |   |   isFound ← checkDiversity(c);
8 |   updatePopulation(pop,c);
9 return pop;
```

TABLE III
ENERGY VALUES OBTAINED BY RUNNING EGA AND WGA TO DEMONSTRATE THE EFFECTIVENESS OF RANDOM-WALK-BASED STAGNATION RECOVERY TECHNIQUE. COLUMN LB-FE PRESENTS THE LOWER BOUND OF FREE ENERGIES

Protein details				EGA			WGA			Time
Seq	Size	Hs	LB-FE	Best	Mean	STD	Best	Mean	STD	mins
H1	48	24	-69	⋆	-67	1.05	⋆	**-68**	0.67	30
H2	48	24	-69	⋆	-67	0.93	⋆	**-68**	0.96	30
H3	48	24	-72	-71	-69	1.29	⋆	**-70**	0.87	30
H4	48	24	-71	⋆	-69	1.31	⋆	**-70**	0.86	30
H5	48	24	-70	⋆	-68	1.10	⋆	**-69**	0.73	30
F90_1	90	50	-168	-159	-144	6.54	**-165**	**-161**	2.53	120
F90_2	90	50	-168	-155	-142	5.84	**-166**	**-161**	2.26	120
F90_3	90	50	-167	-158	-146	5.78	**-165**	**-161**	2.36	120
F90_4	90	50	-168	-158	-146	5.03	**-166**	**-161**	2.42	120
F90_5	90	50	-167	-158	-144	6.43	**-166**	**-162**	2.48	120

⋆ denotes the lower bound of free energy is found.

a) Random-walk: We define a random-walk as a process of exploring feasible solutions in the vicinity of a base solution with minimal changes made to it. Both good and bad solutions are taken into consideration during those changes. The process is repetitive and increases the dissimilarity of the new solution with the base solution.

b) Stagnation in GA: As the search progresses, the GA population becomes richer and richer from one generation to the next. From experimental results, it has been found that after a certain number of generations, a significant portion of the individuals in the population become identical or almost identical. At this situation, the GA operators fail to generate new improved solutions from similar or almost similar parents. Therefore, the search gets stuck and stagnation arises. We evaluate and compare the current best solution with the so-far global-best solution at the end of each generation. When the global-best solution found remains the same for a number of generations (line 20 in Algorithm 8), we consider this stagnation.

c) Random-walk at stagnation: In a stagnation situation, a random-walk technique is applied to perform unconditional pull moves [see Fig. 6(d)] on each conformation of the new population to diversify the population. The pseudocode of the random-walk is shown in Algorithm 12. A local minimum or relative minimum in PSP is encountered when a premature hydrophobic-core is formed. To break the premature hydrophobic-core, we apply a random-walk algorithm.

TABLE IV
FOR SMALL SIZED PROTEINS, THE EXPERIMENTAL RESULTS OF GA VARIANTS, AND THE LS ALGORITHM. THE COLUMNS SIZE AND LBFE PRESENT THE LENGTH OF THE AMINO ACID SEQUENCES AND LOWER BOUND OF FREE ENERGIES, RESPECTIVELY

Protein details				**GA variants energy values**												**State-of-the-art**			*Time*
				RGA		**EGA**		**WGA**		**MGA**		**GAPlus**				**TLS**			
Seq	*Size*	*Hs*	*LBFE*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*STD*		*Best*	*Avg*	*STD*	*(min)*
H1	48	24	-69	-65	-58	⋆	-67	⋆	-68	⋆	-67	⋆	**-69**	0.27		-68	-66	1.13	30
H2	48	24	-69	-63	-57	⋆	-67	⋆	-68	⋆	-68	⋆	**-69**	0.24		=	-65	1.60	30
H3	48	24	-72	-64	-57	-71	-69	⋆	-70	⋆	-71	⋆	**-72**	0.42		-69	-66	1.63	30
H4	48	24	-71	-66	-58	⋆	-69	⋆	-70	⋆	-70	⋆	**-71**	0.14		-70	-65	2.05	30
H5	48	24	-70	-65	-58	⋆	-68	⋆	-69	⋆	-69	⋆	**-70**	0.00		-68	-65	1.70	30
H6	48	24	-70	-63	-58	-69	-67	⋆	-69	⋆	-68	⋆	**-69**	0.45		-69	-66	1.63	30
H7	48	24	-70	-66	-58	⋆	-68	⋆	-69	⋆	-69	⋆	**-70**	0.00		-69	-66	1.88	30
H8	48	24	-69	-62	-55	⋆	-66	⋆	-68	⋆	-68	⋆	**-69**	0.14		-67	-64	1.76	30
H9	48	24	-71	-65	-59	⋆	-69	⋆	-70	⋆	-70	⋆	**-71**	0.27		-68	-66	1.26	30
H10	48	24	-68	-63	-57	⋆	-67	⋆	-67	⋆	-67	⋆	**-68**	0.00		⋆	-65	1.65	30
F90_1	90	50	-168	-133	-120	-159	-144	-165	-161	-165	-159	⋆	**-166**	1.62		-164	-160	2.70	120
F90_2	90	50	-168	-132	-124	-155	-142	-166	-161	-164	-158	⋆	**-165**	1.38		-165	-158	3.78	120
F90_3	90	50	-167	-138	-124	-158	-146	-165	-161	-163	-158	⋆	**-164**	1.37		-165	-159	4.13	120
F90_4	90	50	-168	-141	-124	-158	-146	-166	-161	-165	-160	⋆	**-165**	1.40		-165	-159	3.58	120
F90_5	90	50	-167	-131	-120	-158	-144	-166	-162	-166	-160	⋆	**-166**	0.96		-165	-159	4.12	120

⋆ denotes the lower bound of free energy is found.

TABLE V
FOR LARGE SIZED PROTEINS, THE EXPERIMENTAL RESULTS OF GA VARIANTS, AND THE LS ALGORITHM. THE COLUMNS SIZE AND LBFE PRESENT THE LENGTH OF THE AMINO ACID SEQUENCES AND LOWER BOUND OF FREE ENERGIES, RESPECTIVELY

Protein details				**GA variants energy values**											**State-of-the-art**							*Time*
				RGA		**EGA**		**WGA**		**MGA**		**GAPlus**			**TLS [30]**			**MLS1 [32]**		**MLS2 [33]**		
Seq	*Size*	*Hs*	*LBFE*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*	*STD*	*Best*	*Avg*	*STD*	*Best*	*Avg*	*Best*	*Avg*	*(min)*
S1	135	100	-357	-300	-278	-332	-312	-344	-336	-352	-342	**-355**	**-348**	3.51	-351	-341	7.72	-	-	-	-	120
S2	151	100	-360	-298	-258	-332	-300	-346	-334	-351	-339	**-356**	**-349**	4.64	-355	-343	9.97	-	-	-	-	120
S3	162	100	-367	-290	-250	-322	-297	-347	-334	-347	-335	**-361**	**-348**	6.93	-355	-340	10.62	-	-	-	-	120
S4	164	100	-370	-273	-246	-317	-291	-350	-333	-348	-337	**-364**	**-352**	6.72	-354	-343	9.05	-	-	-	-	120
F180_1	180	100	-378	-256	-229	-309	-280	-330	-315	-347	-333	-351	-341	4.60	-338	-326	5.74	-360	-335	**-361**	**-341**	300
F180_2	180	100	-381	-262	-236	-314	-283	-341	-328	-350	-335	-362	-346	5.24	-345	-333	6.21	-362	-340	**-368**	**-350**	300
F180_3	180	100	-378	-261	-236	-315	-288	-352	-334	-356	-340	-361	-350	5.354	-352	-338	6.86	-357	-343	**-365**	**-355**	300
R1	200	100	-384	-249	-221	-295	-273	-345	-326	-353	-331	**-355**	**-345**	6.074	-332	-318	6.80	-353	-325	-359	-339	300
R2	200	100	-383	-262	-219	-302	-276	-345	-327	-351	-333	**-360**	**-346**	7.11	-337	-324	6.75	-351	-329	-361	-343	300
R3	200	100	-385	-250	-220	-299	-274	-340	-323	-344	-330	**-363**	**-344**	8.21	-339	-323	8.25	-352	-329	-354	-340	300
3MSE	179	84	-323	-198	-178	-221	-206	-271	-249	-278	-268	**-292**	**-278**	6.19	-268	-251	7.32	-278	-255	-288	-271	300
3MR7	189	93	-355	-226	-208	-252	-233	-299	-288	-325	-309	**-330**	**-316**	6.05	-304	-287	5.90	-311	-292	-320	-304	300
3MQZ	215	120	-474	-303	-276	-338	-310	-389	-369	-412	-398	**-427**	**-412**	7.023	-404	-384	7.96	-415	-387	-430	-404	300
3NO6	229	116	-455	-274	-251	-340	-310	-395	-372	-399	-384	**-423**	**-402**	9.59	-390	-372	7.88	-400	-374	-419	-397	480
3NO3	258	122	-494	-288	-250	-359	-316	-394	-375	-406	-387	**-421**	**-404**	8.51	-388	-359	9.79	-379	-361	-406	-376	480
3ON7	279	146	u/k	-359	-315	-428	-389	-482	-458	-494	-473	**-519**	**-490**	12.94	-491	-461	9.32	-499	-463	-514	-476	480

u/k denotes unknown.

In random-walk, we use pull-moves (line 6). During pulling, energy levels and structural diversification are observed to maintain a balance among the two. In a controlled random-walk (line 7), we allow the energy level to change between 5% and 10% and structural change between 10% and 75% of the original. We accept the conformation that has the minimum differential energy level, but has the maximum structural diversity with respect to the original one.

d) Effectiveness of random-walk: In Table III, we present the results obtained from two variants of GA: 1) the EGA and 2) the WGA. From the experimental results, it is clear that the WGA significantly outperforms the EGA. In Table III, we present the global minimum energy (column Best), the average of minimum energies (column Mean), and the standard deviation of the minimum energies (column STD) obtained from 50 different runs of identical settings for all proteins using these two GA variants. For both the cases, the WGA performs better than EGA. This consistent performance demonstrates the effectiveness of the random-walk-based stagnation recovery technique within the GA framework.

5) Housekeeping Between Generations: As mentioned in Section IV-B8, if an improvement has occurred, the nonimproving time tracker and nonimproving generation counter are reset. Conversely, if the current best solution is worse or comparable to the globally best one, the nonimproving time tracker and nonimproving generation counter are incremented. Finally, if the nonimproving time tracker or nonimproving generation counter reach their threshold values, the random-walk process is activated to create diversity in the population.

6) Combined Efforts of all of the Enhancements: In our multistep enhancement, the final outcome is GAPlus. The GAPlus inherits all features of the EGA. Additionally, it includes the hydrophobic-core directed macro-mutation operator and the random-walk-based stagnation recovery technique. The inclusion of the exhaustive generation approach, the random-walk technique, and the macro-mutation operator helps our GAPlus produce the state-of-the-art results in HP based on-lattice PSP.

7) Effectiveness of the Enhanced Features: In Tables IV and V, we present the results obtained from

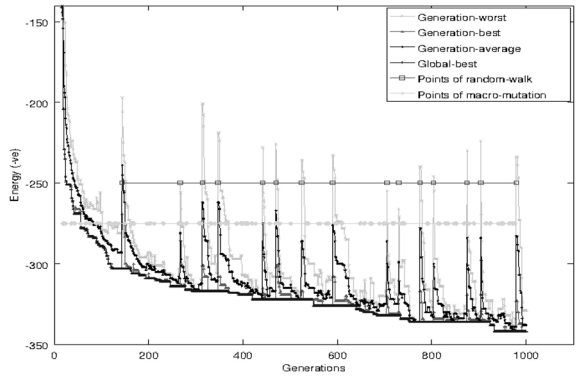


Fig. 9. Activities while GAPlus is in operation. The figure presents the progressive search of a sample run of 1000 generation for Protein R1.

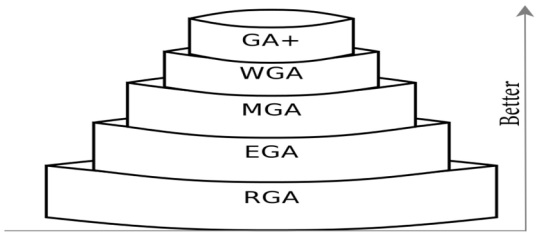


Fig. 10. Performance hierarchy within the GA variants. The higher the position the better the performance.

four variants of GA: 1) EGA; 2) WGA; 3) MGA; and 4) GAPlus. From the experimental results, it is clear that the GAPlus significantly outperforms the other variants. The GAPlus performs better than others in terms of the global minimum free energy and the average free energy for all of the 35 benchmark proteins. This consistent performance demonstrates the effectiveness of the combined efforts of the exhaustive generation, the random-walk, and the macro-mutation operator in GA.

8) Observing GAPlus Features: In Fig. 9, when random-walk occurs the population becomes more diversified. Because of this, the average free energy for the specific generation increases. Similarly, when the macro-mutation applied, the average quality of the individuals improves quickly. Because of this, the average free energy for the specific generation decreases.

D. Performance Hierarchies Amongst the GA Variants

The performance hierarchies amongst the GA variants are presented in Fig. 10. Because of the exhaustive generation technique, the EGA performs better than the RGA. Because of the exhaustive generation in combination with the macro-mutation operator, the MGA performs better than the RGA and the EGA. Because of the exhaustive generation in combination with the random-walk-based stagnation recovery technique, the WGA performs better than the RGA, the EGA, and the MGA. Finally, because of the combination of all three enhancements, the GAPlus performs better than the others.

For further investigation, we perform rigorous analysis on the results obtained from the GA variants and also compare our experimental results with the state-of-the-art approaches for the same models in Section V.

V. Detailed Experimentation and Analyses

In this section, we evaluate the performance of the GAPlus by comparing the results with that of the state-of-the-art approaches. We also perform an extensive experimental study on the GAPlus parameters to find an optimum set of parameters.

A. Experimental Settings

Our implementation is based on the Java programming language. We ran the experiments on the NICTA [70] cluster. This cluster consists of a number of identical Dell PowerEdge R415 computers, each equipped with 2× AMD 6-core Opteron 4184 processors, 2.8 GHz clock speed, 3M L2/6M L3 cache, and 64 GB memory and running Rocks OS (a Linux variant for cluster).

B. Benchmark Proteins Used in GAPlus Evaluation

In our experiment, the protein instances that we use as benchmarks are taken from the literature. The H, F90, S, F180, and R instances are taken from the Peter Clote laboratory website [71]. These benchmarks are used in [30]–[33] to test their algorithms. Additionally, we also use another six large sized sequences that are taken from the critical assessment of PSP (CASP) competition [72]. The CASP9 targets are 3mse, 3mr7, 3mqz, 3no6, 3no3, and 3on7 which are also used in [32] and [33]. To fit in the HP model, the CASP targets are converted into the HP sequences based on the hydrophobic properties (see Section II-C) of the constituent amino acids.

C. State-of-the-Art Methods Used to Evaluate GAPlus

In order to measure the efficiency of our enhanced GA framework (GAPlus), we compare the results with a tabu-guided LS algorithm [30] which is denoted as TLS, a memory-based tabu-guided LS [32] which is denoted as MLS1, an enhanced memory-based tabu-guided LS [33] which is denoted as MLS2, and a population-based memetic algorithm [63] which is denoted as MA in the rest of the sections. We tried to run the algorithm proposed in [31], but, unfortunately for most of the proteins, the program aborted showing the memory related errors. Any effective comparison in this case is therefore not possible.

D. Comparative Analyses for Evaluation

We ran the five variants of GA: 1) RGA; 2) EGA; 3) MGA; 4) WGA; and 5) GAPlus (see Section IV-A for details). For each protein, we run each algorithm 50 times with identical experimental settings and with the time limits specified in Tables IV and V under column Time.

1) Comparing Our Results With the State of the Art Results: Based on the size of the protein sequences, we present the comparative results in two different tables: 1) Table IV presents the small sized proteins (size < 100) and 2) Table V presents the large sized proteins (size > 100). The lower bounds of the free energy (LBFE) values (column LBFE in Tables IV and V) are obtained from [30] and [32]; however,

TABLE VI
COMPARATIVE RESULTS OF GAPLUS WITH THE RECENTLY PUBLISHED REPORTED RESULTS USING MEMETIC ALGORITHM OVER 8 SMALL SIZED PROTEINS

Protein details				Our Approach GAPlus				State-of-the-art MA			
Seq	*Size*	*Hs*	*LBFE*	*Best*	Mean	*STD*	Worst	*Best*	Mean	*STD*	Worst
B1	20	12	-28	**-28**	**-28.00**	±0.00	-28	-23	-23.00	±0.00	-23
B2	24	10	-23	**-23**	**-23.00**	±0.00	-23	**-23**	**-23.00**	±0.00	-23
B3	25	9	-17	**-17**	**-17.00**	±0.00	-17	**-17**	**-17.00**	±0.00	-17
B4	36	16	-38	**-38**	**-38.00**	±0.00	-38	**-38**	**-38.00**	±0.00	-38
B5	48	25	-68	**-68**	**-67.96**	±0.20	-67	**-68**	-66.94	±0.28	-65
B6	50	32	-97	**-99**	**-97.32**	±1.12	-95	-71	-69.15	±0.37	-67
B7	60	32	-98	**-130**	**-129.72**	±0.61	-128	-128	-125.65	±0.34	-124
B8	64	42	-130	**-132**	**-131.00**	±1.12	-129	-128	-124.67	±0.45	-123

there are some unknown values of LBFE for larger sequences (presented as u/k in Table V).

a) Comparing on small sized benchmark proteins: The experimental results in Table IV show that for H instances (size = 48): amongst the GA variants, the EGA is able to achieve the LBFE for 8 proteins out of 10 and the WGA, MGA, and GAPlus are able to hit the LBFE for all 10 proteins. Moreover, the average energy obtained by GAPlus demonstrates that our final version of GA is able to reach native energy levels in almost every single run for H instances. On the other hand, for F90 instances (size = 90): amongst the GA variants, WGA and GAPlus performs better than the state-of-the-art approaches, however, only the GAPlus is able to hit the LBFE for all five proteins. The performances of the MGA and the TLS are very similar.

b) Comparing on large sized benchmark proteins: Notice that the GAPlus, which benefits from all our three novel techniques, clearly outperforms RGA, EGA, MGA, and WGA. Nevertheless, we observe that the results obtained by GAPlus are better than those of TLS and MLS1 with wide margins for all (16 out of 16) benchmark proteins. TLS and MLS1 are also outperformed by WGA, but they outperform EGA and RGA; the results of TLS and MGA are very close. On the other hand, MLS2 produces competitive results for few cases (3 out of 16), but is outperformed by the GAPlus for the rest of the cases.

Furthermore, as shown in Table VI, we ran GAPlus for another set of benchmark proteins taken from [63]. We compared our results with the reported values in [63] produced by an MA. For a set of very small proteins (size < 50), the MA produces competitive results but fails to compete with the GAPlus for a slightly larger set of proteins (size > 50).

2) Relative Improvement: As the predicted energy approaches the LBFE, further improvement becomes more difficult. The relative improvement (RI) explains how close the predicted results (target) are to the LBFEs with respect to the energy values scored by the state-of-the-art (reference) approaches. In Table VII columns RI, we present a comparison of (%) improvements on the average conformation quality. We compare target GAPlus with references RGA, TLS, MLS1, and MLS2. For each protein, the RI of the target (t) with respect to the reference (r) is calculated using (4). We present the RIs only for the proteins having known LBFE

$$\mathsf{RI} = \frac{E_t - E_r}{E_l - E_r} * 100\% \tag{4}$$

TABLE VII
RIS (RI COLUMNS) OF GAPLUS OVER RGA, TLS, AND MLS2. THE VALUES ARE CALCULATED USING THE FORMULA EXPLAINED IN EQUATION (4). COLUMN LBFE PRESENTS THE LOWER BOUND OF FREE ENERGIES

Protein details				GA+ (t)	RGA (r)		TLS [30] (r)		MLS1 [32] (r)		MLS2 [33] (r)	
Seq	*Size*	*Hs*	*LBFE*	*Avg*	*Avg*	*RI*	*Avg*	*RI*	*Avg*	*RI*	*Avg*	*RI*
H1	48	24	-69	-68	-57	92%	-66	67%				
H2	48	24	-69	-68	-57	92%	-65	75%				
H3	48	24	-72	-71	-56	94%	-65	86%				
H4	48	24	-71	-70	-57	93%	-65	83%				
H5	48	24	-70	-70	-57	100%	-65	100%				
F90_1	90	50	-168	-165	-119	94%	-159	67%				
F90_2	90	50	-168	-165	-123	94%	-157	72%				
F90_3	90	50	-167	-164	-124	93%	-158	67%				
F90_4	90	50	-168	-164	-123	91%	-159	56%				
F90_5	90	50	-167	-165	-120	96%	-158	78%				
S1	135	100	-357	-348	-278	89%	-341	44%				
S2	151	100	-360	-349	-258	89%	-343	35%				
S3	162	100	-367	-348	-250	84%	-340	30%				
S4	164	100	-370	-352	-246	85%	-343	33%				
F180_1	180	100	-378	-341	-229	75%	-326	29%	-335	14%	-341	×
F180_2	180	100	-381	-346	-236	76%	-333	27%	-340	15%	-350	×
F180_3	180	100	-378	-350	-236	80%	-338	30%	-343	20%	-355	×
R1	200	100	-384	-345	-221	76%	-318	41%	-325	34%	-339	13%
R2	200	100	-383	-346	-219	77%	-324	37%	-329	31%	-343	8%
R3	200	100	-385	-344	-220	75%	-323	33%	-329	27%	-340	9%
3MSE	179	84	-323	-278	-178	69%	-251	36%	-255	34%	-271	13%
3MR7	189	93	-355	-316	-208	73%	-287	43%	-292	38%	-304	24%
3MQZ	215	120	-474	-412	-276	69%	-384	31%	-387	29%	-404	11%
3NO6	229	116	-455	-402	-251	74%	-372	36%	-374	35%	-397	9%
3NO3	258	122	-494	-404	-250	63%	-359	33%	-361	32%	-476	24%

× denotes "no improvement".

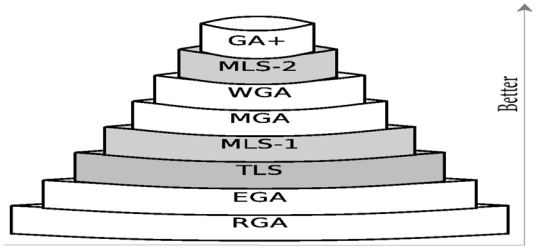


Fig. 11. Performance hierarchy within the GA variants and the state-of-the-art: TLS, MLS1, and MLS2 methods. The higher the position, the better the performance.

where E_t and E_r denote the average energies of the target and the reference algorithms and E_l is the LBFE.

3) Performance Hierarchy: The performance hierarchy amongst the GA variants and the state-of-the-art approaches is presented in Fig. 11. Experimentally, we show that the three variants of GA— the MGA, WGA, and GAPlus—outperform the LS TLS and MLS1. However, another memory-based LS, MLS2, outperforms the MGA and WGA but is outperformed by GAPlus.

4) Search Progress: To demonstrate the search progress, we periodically find the best energy values obtained so far in each run for all approaches. For a given period, we then calculate the average energy values obtained for that period over 50 runs. We used 20 min time intervals to take the average energy values. Fig. 12 presents the chart of search progress with progressive time in terms of average energy values obtained by each algorithm for the protein R1.

We observe that after very small initial progress, the EGA and RGA become flat with almost no improvements.

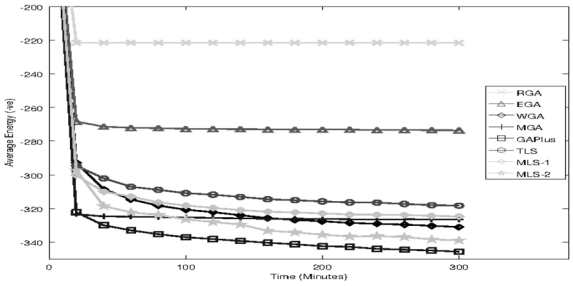


Fig. 12. Search progress of different approaches for Protein R1.

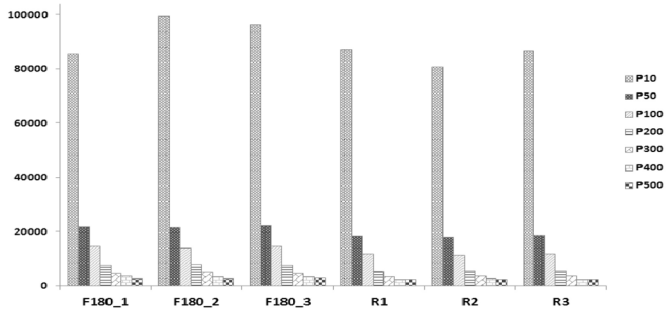


Fig. 13. Effect of population size on number of generations.

The MGA achieves very good progress initially compared to the others, but becomes almost flat later on. The WGA, the TLS, and the MLS1 perform equally initially compared to each other, but later the WGA makes more progress than the TLS and MLS1. The progress of MLS2 is better than that of TLS, MLS1 and other GA variants except GAPlus. Initially, the GAPlus achieves the same progress as the MGA, and later it mostly benefits from the random-walk. The combined positive efforts of the macro-mutation and the random-walk make GAPlus the most efficient conformational search framework. The difference between the performances of WGA and GAPlus remains roughly the same as the initial boosted progress made by the macro-mutation operator.

E. Parameter Tuning for GAPlus

In this section, we work with GAPlus parameters. We experimentally try to work out an optimal population size for GAPlus. We also try to find out the optimal rate of using crossover and mutation operators.

1) Tuning GAPlus Population Size: To observe the effect of population size, we ran GAPlus by setting population sizes 10, 50, 100, 150, 200, 250, 300, 400, and 500. We trace the number of generations, the total number of solutions explored, the total number of stagnations, and the minimum energy level achieved for each population size. We present the average outcomes for the traced values over 50 different runs with further analysis.

a) Population size versus the number of generations: We run GAPlus with different population sizes for proteins F180 and R. We calculate the average number of generations for different population sizes over 50 different runs of identical settings (such as computer, period of running) for each protein. We plot the average number of generations with corresponding population sizes as shown in Fig. 13.

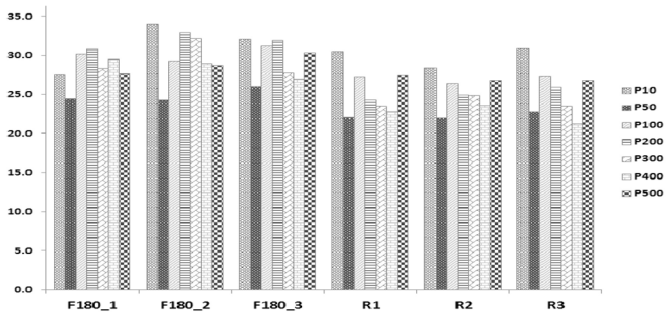


Fig. 14. Effect of population size on explored conformations.

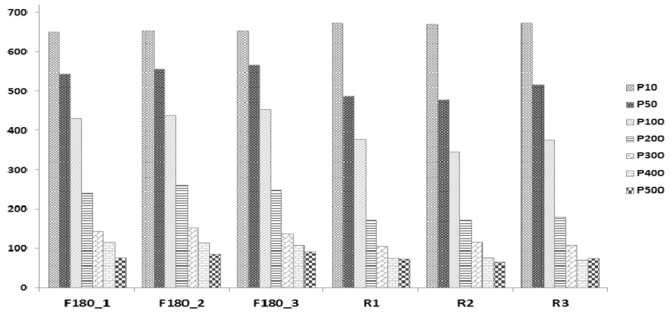


Fig. 15. Effect of population size on average number of stagnations.

The experimental results show that the number of generations is proportionately decreasing when increasing the population sizes. Because of the exhaustive generation approach, the total number of conformations explored in each generation are $O(P \times N)$, where P is the population size and N is the number of amino acids in the protein sequence.

b) Population size versus explored conformations: We plot the average number of explored conformations (in millions) with corresponding population sizes as shown in Fig. 14. The experimental results show that the number of explored conformations does not vary much with the population sizes. Because the exhaustive generation approach in GAPlus does not remember the previously explored solutions, it may re-explore early explored conformations throughout the period of running. Nevertheless, it is notable that the number of explored conformations is consistently at a minimum for population size 50.

c) Population size versus the number of stagnations: We plot the average number of stagnations with corresponding population sizes as shown in Fig. 15. The experimental results show that the number of occurred stagnations is proportionately decreasing with increasing the population sizes. The result is expected because at stagnation random-walk is applied and the population is diversified. Therefore for larger population size, it takes more time to reach the stagnation situation by intensifying the population again. The frequency of occurring stagnation decreases while the population size increases.

d) Population sizes versus free energy levels: We plot the average energy values with corresponding population sizes as shown in Fig. 16. The experimental results show that the levels of average free energies do not vary much with the population sizes. However, it is notable that for population

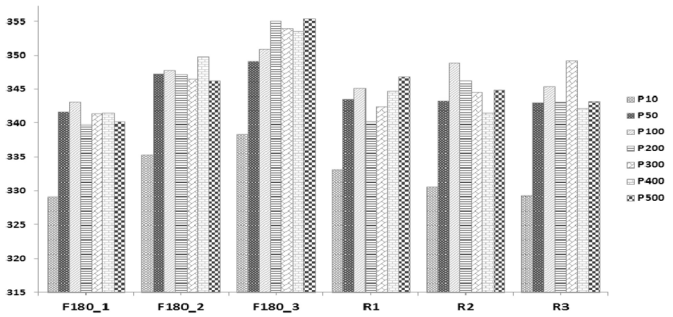


Fig. 16. Effect of population size on average energy values.

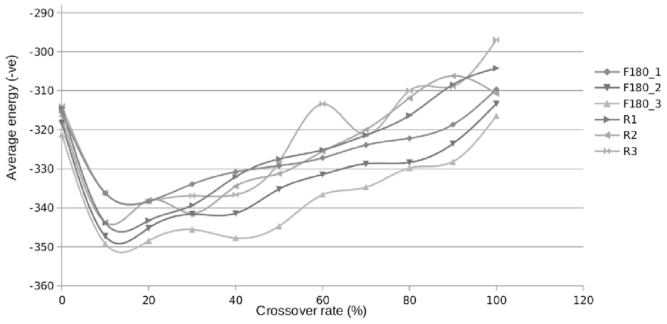


Fig. 17. Average energy values corresponding to crossover rates.

size 10 (very small), the average energy value for each protein is consistently the minimum amongst the population sizes. Because, for a very small population size, the diversification amongst the individuals is narrow in comparison to other reasonably large population sizes.

In conclusion, observing the effect of population sizes, it is notable that the number of generations and the number of stagnation situations decrease proportionately with increasing population sizes; the impacts on the number of explored conformations with the change of population size is not significant; and the impact of population size on the lower energy level is interesting—for population size 10, the GAPlus performs poorly (due to insufficient diversity) for each protein, however, for other population sizes, the results are very close for each protein.

2) Profiling GAPlus Operators: In this section, we experimentally tried to work out an optimal crossover (or mutation) rate for GAPlus. We run GAPlus for crossover rates 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% (corresponding mutation rates are 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%, and 0%). We have observed the variations on average energy levels achieved for different crossover or mutation rates in the experimental results.

We further trace some characteristics during operation: in the case of mutation operators, we use all primitive mutation operators such as rotation, diagonal moves, pull moves, and tilt moves (see Fig. 6) and present the combined results under the mutation operator. All primitive mutation operators are chosen with equal selection probability.

a) Effects of crossover rates on energy levels: From our experimental results on 11 different crossover rates, we plot a chart of average energy levels with the crossover rates as shown in Fig. 17. The chart shows a clear impact of crossover rates on average energy levels. The smaller sizes of crossover rates are more effective than that of the larger sizes. In our experiment, the crossover rate in the range of $10\%-20\%$ (i.e., mutation rates: $80\%-90\%$) produces better results in GAPlus.

b) Effects of crossover rates on other factors: Other GA factors such as operator selection, success of operator application, impact of the applied operator, and total usages of any operator are traced and the summarized results are presented in Table VIII. For each operator, we observed the number of times an operator is selected to apply (column Attempt), how many times the attempts are successful (column Succeed), how many times it improves locally (column Gain), and how much time the operator consumed throughout the life of the run (column Usage). The presented averages are the outcomes over 30 different runs. The experiments are conducted with the population size 100.

TABLE VIII
EFFECT OF THE APPLICATIONS OF OPERATORS IN THEIR VARIOUS RATES. THE COLUMN XO DENOTES CROSSOVER

	XO	**Crossover Operators**				**Mutation Operators**				*Avg*
Seq	*Rate*	*Attempt*	*Succeed*	*Gain*	*Usage*	*Attempt*	*Succeed*	*Gain*	*Usage*	*energy*
[size]	*[%]*	*[mill]*	*[%]*	*[%]*	*[%]*	*[mill]*	*[%]*	*[%]*	*[%]*	level
	0	0	0	0	0	89.67	31.89	1.08	100	-333.4
	10	1.5	32.53	30.83	4	84.65	31.15	0.72	96	**-348.1**
	20	4.17	48.34	19.14	16	71.54	30.93	0.72	84	-348.0
	30	9.14	54.6	8.66	38	51.1	30.99	0.74	58	-346.3
	40	14.18	58.15	5.18	60	36.07	31.24	0.72	40	-347.0
S1	50	14.86	60.75	4.17	74	21.45	30.94	0.83	26	-342.8
[135]	60	17.33	63.02	3.57	83	14.66	31	0.82	17	-344.1
	70	19.3	63.2	3.37	89	9.8	31.02	0.87	11	-339.6
	80	18.09	64.8	3.66	94	5.16	30.63	1.02	6	-338.8
	90	18.9	67.43	5.13	97	2.37	29.97	1.09	3	-336.3
	100	19.54	70.61	10.54	100	0.11	0	0	0	-330.4
	0	0	0	0	0	76.77	29.46	1.04	100	-315.5
	10	1.12	26.54	33.78	4	64.05	28.75	0.7	96	-336.2
	20	2.7	43.24	30.22	13	57.43	28.68	0.68	87	**-338.3**
	30	6.03	50.61	14.95	34	43.5	28.86	0.71	66	-333.9
	40	8.56	54.39	9.61	54	28.21	29.14	0.79	46	-330.8
F1	50	11.33	56.4	6.84	70	19.92	28.46	0.75	30	-329.3
[180]	60	12.9	58.43	6.05	77	12.87	30	0.84	19	-327.3
	70	12.44	60.99	5.95	87	7.53	29.29	0.95	13	-324.0
	80	14.04	62.62	5.93	93	4.51	29.89	0.89	7	-322.2
	90	14.77	65.54	7.32	97	2.14	29.95	1.09	3	-318.7
	100	12.87	69.6	10.01	100	0.08	0	0	0	-309.65
	0	0	0	0	0	54.29	29.13	1.01	100	-314.4
	10	0.85	30.89	42.91	4	52.85	28.79	0.57	96	**-343.9**
	20	2.34	42.86	30.29	13	51.29	28.02	0.56	87	-343.3
	30	5.36	49.74	13.98	33	39.06	28.37	0.57	67	-339.3
	40	8.27	53.81	8.73	54	26.21	28.6	0.63	46	-332.0
R1	50	10.18	57.54	6.99	70	17.52	29.26	0.72	30	-327.5
[200]	60	11.13	58.92	5.82	80	11.26	29.26	0.75	20	-325.2
	70	12.18	61.86	5.16	88	7.1	29.22	0.8	12	-321.4
	80	11.73	66.03	5.47	93	3.88	30.65	0.86	7	-316.4
	90	12.68	66.66	6.02	97	1.82	30.54	0.94	3	-308.5
	100	12.01	71.57	7.11	100	0.07	0	0	0	-304.3

Table VIII shows that successful attempts and local energy gain on successful attempts are significantly higher for crossover in comparison to mutation. However, from the experimental results, it is observed that these local energy gains have little or no impact on the global minimum energy. Another important observation from the experimental results is that crossover consumes almost double operational-time than mutation operators. The time consumption of the crossover operator is reasonable because the crossover operator generates two resultant solutions but mutation generates one.

Observing the experimental results obtained by using different crossover rates (and therefore different mutation rates) in GAPlus, we can conclude that the small crossover

rates (i.e., high mutation rates) are better for finding lower energy structures on FCC lattice models.

F. Summary of the Study

After implementing the proposed new features within GA framework and studying the GA parameters, we have the following finding:

1) *Feature Enhancement:* The exhaustiveness of individual generation helps improve over random individual generation. The macro-mutation operator and the random-walk technique separately obtain more improvements, with their combination obtaining even further improvements.
2) *Parameter Tuning:* Using any population size > 10, the performance of GAPlus does not vary much. The crossover consumes more operational time in comparison to mutation and leads to poorer performance when high crossover rates ($>20\%$) are used in GAPlus.

VI. Discussion

The proposed GAPlus framework is initially tested on lattice-based PSP. The problem is chosen because it is a hard combinatorial optimization problem with an astronomically large search space. However, the enhanced features could easily be adopted by other branches of optimization problems such as scheduling, traveling salesman, and vehicle routing. The following discussion would be the future directions of this paper.

1) The exhaustive generation strategy implemented in GAPlus is not a complete or full exhaustive sampling. In contrast, this approach selects successively one amino acid at a time and explores exhaustively in the vicinity of the selected amino acid. It does not sample all possible combinations considering all the amino acids at a time (see Section IV-C2 for details). This strategy might be adopted in any optimization problem that uses some sort of randomness in exploring neighbor solutions.
2) The macro-mutation operator quickly improves an individual by applying multiple mutations in a single pass. This operator could be applied in other classes of optimization problems to improve a solution either by considering a segment of the solution or by considering the solution as a whole. This operator is guided by a separate heuristic—distance from the HCC—other than the objective function (energy function for PSP). Thus, designing a problem specific heuristic is a key to implement this operator.
3) The random restarting procedure could replace the random-walk technique regardless of the class of optimization problem. Handling stagnation or dealing with local minima is an integral part of an optimization algorithm. Random-restart is an effective strategy in such situations; however, we experimentally showed that the random-walk is more effective for the GAPlus framework.
4) By encoding the conformation with angular coordinates (ϕ and ψ), the GAPlus might be applied in high-resolution PSP. While the minimizing energy function is highly complex (such as molecular dynamics), a simple guidance heuristic (such as hydrophobic property or exposed surface area) could be used to guide the macro-mutation operator. Within GAPlus framework, the macro-mutation operator could be applied optimizing the segments of secondary structures (α—helix and β—sheet).
5) More realistic models with complicated energy functions increase the complexity of the problem, however, our approach has a macro-mutation operator that performs as a local optimizer. Based on this, our approach can easily divide the whole optimization process into two stages guided by two energy models with different complexities. The macro-mutation operator can be guided by simpler energy models such as distance from hydrophobic core, exposed surface area, hydrophobicity of amino acids, hydropathy index of the amino acids, and so on. Conversely, the main objective function can be more realistic such as molecular dynamics-based energy models. This two-stage optimization will reduce the overall computational complexities. As a result, our framework has a good chance to succeed in more realistic models even for large sized proteins.
6) The on-lattice PSP problem can be applied directly to the other classes of optimization problems such as computer organization [73], very large scale integration design [74], and network topology designing [75] where nodes or processors are connected in a daisy chain but a subset of those need to be topologically compact or closer to have efficient dedicated connections. The PSP application can also be extended to solve quadratic assignment problems using lattice model encoding in genetic algorithms [76], solving job shop scheduling problems in a multiagent environment [77], and so on.

VII. Conclusion

In this paper, we presented five variants of GAs that individually and in a combined way use three different enhancement techniques: 1) an exhaustive generation approach; 2) a novel hydrophobic-core directed macro-mutation operator; and 3) a random-walk-based stagnation recovery technique. We found that our final algorithm, GAPlus, which uses a combination of all the enhancement techniques, significantly outperforms all current approaches to simplified PSP. We also showed that our algorithm is robust enough to produce very similar results under different parameter settings. In the future, we intend to apply GAPlus in high resolution PSP. In particular, we want to configure GAPlus within the Rosetta Codebase so that it could be used as a separate protocol in place of the MC simulation method. We also aim to test GAPlus for other branches of optimization such as scheduling, traveling salesman, and vehicle routing.

Acknowledgment

The first author would like to thankfully acknowledge M. A. H. Newton and D. N. Pham for their constructive feedback at the preliminary stage of this work. The authors would also like to express their sincere thanks and appreciation to N. Trieber for his excellent proofreading and editing service.

REFERENCES

[1] H. J. Morowitz, *Energy Flow in Biology*. New York, NY, USA: Academic Press, 1968.

[2] A. H. Stouthamer, "A theoretical study on the amount of ATP required for synthesis of microbial cell material," *Antonie Van Leeuwenhoek*, vol. 39, no. 1, pp. 545–565, 1973.

[3] B. Alberts *et al.*, "The shape and structure of proteins," in *Molecular Biology of the Cell*, 4th ed. New York, NY, USA: Garland Sci., 2002.

[4] N. Horton and M. Lewis, "Calculation of the free energy of association for protein complexes," *Protein Sci. Pub. Protein Soc.*, vol. 1, no. 1, pp. 169–181, 1992.

[5] O. D. King, A. D. Gitler, and J. Shorter, "The tip of the iceberg: RNA-binding proteins with prion-like domains in neurodegenerative disease," *Brain Res.*, vol. 1462, pp. 61–80, Jun. 2012.

[6] A. Smith, "Protein misfolding," *Nat. Rev. Drug Disc.*, vol. 426, no. 6968, pp. 78–102, Dec. 2003.

[7] C. M. Dobson, "Protein folding and misfolding," *Nature*, vol. 426, no. 6968, pp. 884–890, 2003.

[8] F. Chiti and C. M. Dobson, "Protein misfolding, functional amyloid, and human disease," *Annu. Rev. Biochem.*, vol. 75, no. 1, pp. 333–366, 2006.

[9] M. Jucker and L. C. Walker, "Self-propagation of pathogenic protein aggregates in neurodegenerative diseases," *Nature*, vol. 501, no. 7465, pp. 45–51, 2013.

[10] P. Veerapandian, Ed., *Structure-Based Drug Design*. New York, NY, USA: Marcel Dekker, 1997.

[11] A. Breda, N. F. Valadares, O. N. de Souza, and R. C. Garratt, "Protein structure, modelling and applications," in *Bioinformatics in Tropical Disease Research: A Practical and Case-Study Approach*. Bethesda, MD, USA: Nat. Center Biotechnol. Inf., 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK6824/

[12] P. Kast and D. Hilvert, "3D structural information as a guide to protein engineering using genetic selection," *Curr. Opinion Struct. Biol.*, vol. 7, no. 4, pp. 470–479, 1997.

[13] S. M. Berry and Y. Lu, *Protein Structure Design and Engineering*. New York, NY, USA: Wiley, 2001.

[14] H. Lilie, "Designer proteins in biotechnology," *EMBO Rep.*, vol. 4, no. 4, pp. 346–351, 2006.

[15] R. Jaenicke and R. Sterner, "Protein design at the crossroads of biotechnology, chemistry, theory, and evolution," *Angew. Chem. Int. Ed.*, vol. 42, no. 2, pp. 140–142, 2003.

[16] A. Yonath, "X-ray crystallography at the heart of life science," *Curr. Opin. Struct. Biol.*, vol. 21, no. 5, pp. 622–626, 2011.

[17] The Science Editorial, "So much more to know," *Science*, vol. 309, no. 5731, pp. 78–102, Jul. 2005.

[18] A. Leaver-Fay *et al.*, *ROSETTA3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules* (Method Enzymol), vol. 487. New York, NY, USA: Academic Press, 2011, pp. 545–574.

[19] K. E. Kemege *et al.*, "Ab initio structural modeling of and experimental validation for Chlamydia trachomatis protein CT296 reveal structural similarity to Fe(II) 2-oxoglutarate-dependent enzymes," *J. Bacteriol.*, vol. 193, no. 23, pp. 6517–6528, 2011.

[20] D. Xu and Y. Zhang, "Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field," *Proteins*, vol. 80, no. 7, pp. 1715–1735, Apr. 2012.

[21] K. Molloy, S. Saleh, and A. Shehu, "Probabilistic search and energy guidance for biased decoy sampling in *ab initio* protein structure prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 10, no. 5, pp. 1162–1175, Sep. 2013.

[22] B. Olson and A. Shehu, "An evolutionary-inspired algorithm to guide stochastic search for near-native protein conformations with multiobjective analysis," in *Proc. AAAIW*, Bellevue, WA, USA, 2013, pp. 32–37.

[23] P. Bradley, K. M. S. Misura, and D. Baker, "Toward high-resolution *de novo* structure prediction for small proteins," *Science*, vol. 309, no. 5742, pp. 1868–1871, 2005.

[24] D. E. Kim, B. Blum, P. Bradley, and D. Baker, "Sampling bottlenecks in *de novo* protein structure prediction," *J. Mol. Biol.*, vol. 393, no. 1, pp. 249–260, 2009.

[25] J. Zhang, "Protein-length distributions for the three domains of life," *Trends Genet.*, vol. 16, no. 3, pp. 107–109, 2000.

[26] L. Brocchieri and S. Karlin, "Protein length in eukaryotic and prokaryotic proteomes," *Nucleic Acids Res.*, vol. 33, no. 10, pp. 3390–3400, 2005.

[27] T. B. Higgs, B. Stantic, M. T. Hoque, and A. Sattar, "Applying feature-based resampling to protein structure prediction," in *Proc. BiCoB*, Las Vegas, NV, USA, 2012, pp. 239–244.

[28] C. Levinthal, "Are there pathways for protein folding?" *J. Chim. Phys.*, vol. 65, no. 1, pp. 44–45, 1968.

[29] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, no. 4096, pp. 223–230, 1973.

[30] M. Cebrián, I. Dotú, P. Van Hentenryck, and P. Clote, "Protein structure prediction on the face centered cubic lattice by local search," in *Proc. 23rd Nat. Conf. Artif. Intell.*, vol. 1. Chicago, IL, USA, 2008, pp. 241–246.

[31] I. Dotu, M. Cebrián, P. Van Hentenryck, and P. Clote, "On lattice protein structure prediction revisited," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 8, no. 6, pp. 1620–1632, Nov./Dec. 2011.

[32] S. Shatabda, M. A. H. Newton, D. N. Pham, and A. Sattar, "Memory-based local search for simplified protein structure prediction," in *Proc. ACM Conf. Bioinformat. Comput. Biol. Biomed. (ACM-BCB)*, Orlando, FL, USA, 2012, pp. 345–352.

[33] S. Shatabda, M. A. H. Newton, M. A. Rashid, D. Pham, and A. Sattar, "The road not taken: Retreat and diverge in local search for simplified protein structure prediction," *BMC Bioinformat.*, vol. 14, no. S-2, p. S19, Jan. 2013.

[34] M. T. Hoque, M. Chetty, and A. Sattar, "Protein folding prediction in 3D FCC HP lattice model using genetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 4138–4145.

[35] H.-J. Böckenhauer, A. Z. M. D. Ullah, L. Kapsokalivas, and K. Steinhöfel, "A local move set for protein folding in triangular lattice models," in *Proc. WABI*, Karlsruhe, Germany, 2008, pp. 369–381.

[36] M. A. Rashid, M. T. Hoque, M. A. H. Newton, D. N. Pham, and A. Sattar, "A new genetic algorithm for simplified protein structure prediction," in *Proc. 25th Australasian Joint Conf. Artif. Intell. (AI'12)*, Sydney, NSW, Australia, Dec. 2012, pp. 107–119.

[37] T. Schwede, J. Kopp, N. Guex, and M. C. Peitsch, "SWISS-MODEL: An automated protein homology-modeling server," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3381–3385, 2003.

[38] D. Higgins and W. Taylor, *Bioinformatics: Sequence, Structure and Databanks*, 1st ed. Oxford, U.K.: Oxford Univ. Press, 2001.

[39] J. Skolnick, D. Kihara, and Y. Zhang, "Development and large scale benchmark testing of the PROSPECTOR_3 threading algorithm," *Proteins*, vol. 56, no. 3, pp. 502–518, 2004.

[40] H. Zhou and Y. Zhou, "Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments," *Proteins*, vol. 58, no. 2, pp. 321–328, 2005.

[41] S. Wu and Y. Zhang, "MUSTER: Improving protein sequence profile-profile alignments by using multiple sources of structure information," *Proteins*, vol. 72, no. 2, pp. 547–556, 2008.

[42] L. A. Kelley and M. J. E. Sternberg, "Protein structure prediction on the Web: A case study using the Phyre server," *Nat. Protoc.*, vol. 4, no. 3, pp. 363–371, 2009.

[43] S. Wu, J. Skolnick, and Y. Zhang, "Ab initio modeling of small proteins by iterative TASSER simulations," *BMC Biol.*, vol. 5, no. 1, pp. 17–26, 2007.

[44] T. C. Hales, "A proof of the Kepler conjecture," *Ann. Math.*, vol. 162, no. 3, pp. 1065–1185, 2005.

[45] 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.

[46] C. Thachuk, A. Shmygelska, and H. H. Hoos, "A replica exchange Monte Carlo algorithm for protein folding in the HP model," *BMC Bioinform.*, vol. 8, no. 1, p. 342, 2007.

[47] A.-A. Tantar, N. Melab, and E.-G. Talbi, "A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction," *Soft Comput. Fusion Found. Methodol. Appl.*, vol. 12, no. 12, pp. 1185–1198, 2008.

[48] R. Unger and J. Moult, "A genetic algorithm for 3D protein folding simulations," in *Proc. 5th Int. Conf. Genet. Algorithms*, San Mateo, CA, USA, 1993, pp. 581–588.

[49] M. T. Hoque, "Genetic algorithm for *ab initio* protein structure prediction based on low resolution models," Ph.D. dissertation, Gippsland School Inform. Technol., Monash Univ., Melbourne, VIC, Australia, Sep. 2007.

[50] G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher, "Human-guided tabu search," in *Proc. 18th Nat. Conf. Artif. Intell. (AAAI)*, Edmonton, AB, Canada, 2002, pp. 41–47.

[51] C. Blum, "Ant colony optimization: Introduction and recent trends," *Phys. Life Rev.*, vol. 2, no. 4, pp. 353–373, 2005.

[52] I. Kondov and R. Berlich, "Protein structure prediction using particle swarm optimization and a distributed parallel approach," in *Proc. ACM Workshop Biol. Inspired Algorithms Distrib. Syst.*, Karlsruhe, Germany, 2011, pp. 35–42.
[53] N. Mansour, F. Kanj, and H. Khachfe, "Particle swarm optimization approach for protein structure prediction in the 3D HP model," *Interdiscipl. Sci.*, vol. 4, no. 3, pp. 190–200, Sep. 2012.
[54] 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.
[55] M. Mann, S. Will, and R. Backofen, "CPSP-tools—Exact and complete algorithms for high-throughput 3D lattice protein studies," *BMC Bioinform.*, vol. 9, no. 1, p. 230, 2008.
[56] M. T. Hoque, M. Chetty, A. Lewis, and A. Sattar, "Twin removal in genetic algorithms for protein structure prediction using low-resolution model," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 8, no. 1, pp. 234–245, 2011.
[57] L. Kapsokalivas, X. Gan, A. A. Albrecht, and K. Steinhöfel, "Population-based local search for protein folding simulation in the MJ energy model and cubic lattices," *Comput. Biol. Chem.*, vol. 33, no. 4, pp. 283–294, Aug. 2009.
[58] A. D. Ullah and K. Steinhöfel, "A hybrid approach to protein folding problem integrating constraint programming with local search," *BMC Bioinform.*, vol. 11, no. 1, p. S39, 2010.
[59] A. Dal Palù, A. Dovier, F. Fogolari, and E. Pontelli, "Exploring protein fragment assembly using CLP," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, Barcelona, Spain, 2011, pp. 2590–2595.
[60] S. Shatabda, M. A. H. Newton, and A. Sattar, "Mixed heuristic local search for protein structure prediction," in *Proc. 27th AAAI Conf. Artif. Intell.*, Bellevue, WA, USA, 2013, pp. 876–882.
[61] S. Miyazawa and R. L. Jernigan, "Estimation of effective interresidue contact energies from protein crystal structures: Quasi-chemical approximation," *Macromolecules*, vol. 18, no. 3, pp. 534–552, 1985.
[62] M. Berrera, H. Molinari, and F. Fogolari, "Amino acid empirical contact energy definitions for fold recognition in the space of contact maps," *BMC Bioinform.*, vol. 4, no. 1, p. 8, 2003.
[63] M. K. Islam and M. Chetty, "Clustered memetic algorithm with local heuristics for ab initio protein structure prediction," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 558–576, Aug. 2013.
[64] K. Yue and K. A. Dill, "Sequence-structure relationships in proteins and copolymers," *Phys. Rev. E*, vol. 48, no. 3, pp. 2267–2278, 1993.
[65] N. Lesh, M. Mitzenmacher, and S. Whitesides, "A complete and effective move set for simplified protein folding," in *Res. Comput. Molecular Biol.*, Berlin, Germany, 2003, pp. 188–195.
[66] M. Rocha and J. Neves, "Preventing premature convergence to local optima in genetic algorithms via random offspring generation," in *Proc. Int. Conf. Ind. Eng. Appl. Artif. Intell. Expert Syst.*, Cairo, Egypt, 1999, pp. 127–136.
[67] T. F. Gonzalez, *Handbook of Approximation Algorithms and Metaheuristics*. Boca Raton, FL, USA: Chapman & Hall/CRC, 2007, ch. 14, p. 12.
[68] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. Hershey, PA, USA: Inf. Sci. Ref., 2010, p. 83.
[69] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004, ch. 3, p. 86.
[70] NICTA. *National ICT Australia Research Centre of Excellence*. [Online]. Available: www.nicta.com.au, accessed Feb. 4, 2016.
[71] P. Clote. *Peter Clote Laboratory Website*. [Online]. Available: bioinformatics.bc.edu/clotelab/FCCproteinStructure/, accessed Nov. 23, 2013.
[72] CASP. *The Critical Assessment of Protein Structure Prediction*. [Online]. Available: http://predictioncenter.org, accessed Feb. 4, 2016.
[73] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. Amsterdam, The Netherlands: Elsevier, 2013, p. 800.
[74] S. D. Kugelmass, R. Squier, and K. Steiglitz, "Performance of VLSI engines for lattice computations," *Complex Syst.*, vol. 1, no. 5, pp. 939–965, 1987.
[75] J. Degila and B. Sanso, "Topological design optimization of a yottabit-per-second lattice network," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1613–1625, Nov. 2004.
[76] R. Cheng and M. Gen, *Genetic Algorithms and Engineering Design*. New York, NY, USA: Wiley, 2007.
[77] X. Duan, J. Liu, L. Zhang, and L. Jiao, *Multi-Objective Job Shop Scheduling Based on Multiagent Evolutionary Algorithm*, Berlin, Germany: Springer, 2010, pp. 543–552.

Mahmood A. Rashid received the B.Sc. degree in mechanical engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 1999, the M.Sc. degree in computer science from IBAIS University, Dhaka, Bangladesh, in 2005, and the Ph.D. degree in computer science from Griffith University, Brisbane, QLD, Australia, in 2014.

He was a Post-Doctoral Fellow with the Department of Computer and Information Science, University of Massachusetts Dartmouth, Dartmouth, MA, USA. Since 2014, he has been an Adjunct Research Fellow with the Institute for Integrated and Intelligent Systems, Griffith University. From 2000 to 2009, he was an IT Specialist with Information Technology Industries. He was a Graduate Researcher with NICTA, Sydney, NSW, Australia. His current research interests include optimization algorithms, the parallel and distributed computing, the search guidance heuristics, and the computational biology.

Dr. Rashid is a fellow with the School of Computing, Information and Mathematical Sciences, University of the South Pacific, Suva, Fiji.

Firas Khatib received the bachelor's degree in applied mathematics from the University of California at Berkeley, Berkeley, CA, USA, in 2001, and the Ph.D. degree in bioinformatics from the University of California at Santa Cruz, Santa Cruz, CA, USA, in 2008.

He was a Senior Fellow with the Molecular Engineering & Sciences Institute, University of Washington, Seattle, WA, USA. He is an Assistant Professor with the Computer and Information Science Department, University of Massachusetts Dartmouth, Dartmouth, MA, USA.

Prof. Khatib was a recipient of the Post-Doctoral Research Fellowship in Biology from the National Science Foundation to work on the protein-folding game Foldit. The long term goal of this project is to utilize the combined power of humans and computers in order to build accurate models of disease-related proteins.

Md Tamjidul Hoque received the Ph.D. degree in information technology from Monash University, Melbourne, VIC, Australia, in 2008.

He is currently an Assistant Professor with the Computer Science Department, University of New Orleans, New Orleans, LA, USA. From 2011 to 2012, he was a Post-Doctoral Fellow with Indiana University–Purdue University Indianapolis, Indianapolis, IN, USA. From 2007 to 2011, he was a Research Fellow with Griffith University, Brisbane, QLD, Australia. He also developed efficient algorithms for high-throughput screening and high content image analysis. He was a Deputy General Manager and the Head of IT in a large industrial-group from 1999 to 2004, where he developed a number of commercial software for the industries. His current research interests include machine learning, evolutionary computation and artificial intelligence, applying toward hard optimization problems especially for biology and bioinformatics such as protein-structure-prediction, disorder-predictor, and energy function.

Abdul Sattar received the Ph.D. degree in computer science (Artificial Intelligence) from the University of Alberta, Edmonton, AB, Canada, in 1990.

He is the Founding Director of the Institute for Integrated and Intelligent Systems and a Professor of Computer Science and Artificial Intelligence with Griffith University, Brisbane, QLD, Australia. He is also one of the Research Leaders with the NICTA Queensland Research Laboratory. Since 1992, he has been an Academic Staff Member with Griffith University, where he was a Lecturer from 1992 to 1995, a Senior Lecturer from 1996 to 1999, and a Professor since 2000 with the School of Information and Communication Technology. Prior to joining Griffith University, he was a Lecturer of Physics in Rajasthan, India, from 1980 to 1982, and a Research Scholar with Jawaharlal Nehru University, New Delhi, India, from 1982 to 1985, the University of Waterloo, Waterloo, ON, Canada, from 1985 to 1987, and the University of Alberta, Edmonton, AB, Canada, from 1987 to 1991.