Solving Lattice Protein Folding Problems by Discrete Particle Swarm Optimization

Jing Xiao
School of Computer Science, South China Normal University, Guangzhou 510631, China
State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China
Email: xiaojing@scnu.edu.cn

Liang-Ping Li
Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China
Email: liangpingli@gmail.com

Xiao-Min Hu
School of Public Health, Sun Yat-sen University, Guangzhou 510080, China
Email: huxiaom6@mail.sysu.edu.cn

Abstract—**Using computer programs to predict protein structures from a mass of protein sequences is promising for discovering the relationship between the protein construction and their functions. In the area of computational protein structure analysis, the hydrophobic-polar (HP) model is one of the most commonly applied models. The protein folding problem based on HP model has been shown as NP-hard, to handle such an NP-hard problem, this paper proposes a discrete particle swarm optimization algorithm ($DPSO_{HP}$) to solve various 2D and 3D HP lattice models-based protein folding problems. The discrete particle swarm optimization method used in $DPSO_{HP}$ is based on the set concept and the possibility theory from a set-based PSO (S-PSO). A selection strategy incorporating heuristic information and possibilities is adopted in $DPSO_{HP}$. A particle's positions in the algorithm are defined as a set of elements and the velocities of a particle are defined as a set of elements associated with possibilities. The experimental results on a series of 2D and 3D protein sequences show that $DPSO_{HP}$ is promising and performs better than various competitive state-of-the-art evolutionary algorithms.**

Index Terms—**Bioinformatics, Computational intelligence, Discrete particle swarm optimization, Hydrophobic-polar (HP) model, Lattice protein folding**

I. Introduction

The structure of a protein has a direct impact on its expressive functionalities in nature. Although protein sequencing has become easier in bioinformatics, the relationship between the protein sequence and its structure is still an open problem. Experimental methods such as X-ray crystallography and nuclear magnetic resonance (NMR) for finding the structure of a protein are still quite expensive and time-consuming. As the development of molecular dynamics, several theories have been proposed to explain the conformation of a protein based on its sequences [1, 2]. One of the most influential findings is that given a protein sequence, the structure of the protein is unique under suitable conditions and forms a minimum free energy (MFE) conformation [1]. This observation has been widely accepted and acts as the basis for the modern prediction methods of protein structures based on its sequences [3, 4, 5].

Several computational models for constructing the MFE conformations have been proposed, such as the hydrophobic-hydrophilic (HP) square lattice model [2], toy model [6], functional model [7], and HP side chain model [8], etc. The HP model is the most simplified model [3, 4, 5, 9, 10]. In HP model, all amino acids in a protein are classified as hydrophobic amino acids (H) and hydrophilic polar amino acids (P). Hydrophobic amino acids have aliphatic side chains and avoid water. So they generally form a protein core in the middle of the protein. In contrast, hydrophilic amino acids, also termed as polar amino acids, have an affinity with the solvent and tend to remain in the outer surface of the protein. According to the considered dimensions, an HP model can be in two dimensions (2D) or three dimensions (3D). A two dimensional square lattice is simulated in the 2D HP model, whereas a three dimensional cubic lattice is used in a 3D HP model. The protein folds in the lattice with each amino acid being placed in one cell of the lattice. Even though HP model is simple, the protein folding problem (PFP) based on HP model has been proved to be NP-hard [11, 12].

Computational intelligence (CI) algorithms are promising for solving the PFP with HP model. They are stochastic and heuristics-based methods and have been successfully applied in various NP-hard problems. For the PFP with HP model, traditional CI methods such as Monte Carlo-based algorithms [13, 14, 15], genetic algorithms (GAs) [16, 17, 18, 19, 20, 21], immune algorithms (IMs) [3, 22], differential evolution [23, 24], ant colony optimization (ACO) algorithms [4, 25, 26, 27], and estimation of distribution algorithms (EDAs) [5, 28] have been proposed. Among the above algorithms, the flexible ant colony (FAC) algorithm [4] used a special heuristic method and a path retrieval mechanism. The EDAs proposed in [5] used explicit probability models to construct protein conformations. For example, the Markov-EDA (MK-EDA) adopted a Markov probabilistic model and the Tree-EDA was based on a tree probabilistic model. The MT-EDA used a mixture of trees and the experimental results showed that it performed better than Tree-EDA for some protein sequences. Although there are many CI algorithms that have been proposed to solve the PFP, the performance is still unsatisfactory.

In this paper, a particle swarm optimization (PSO) is proposed for solving the PFP. PSO [29] is a swarm-based CI algorithm and has been proven to obtain good performance in continuous domains [30, 31, 32]. A discrete roulette-based PSO is employed in PFP in [33] and the experiments are conducted on 6 proteins. In [34] the authors employed a set-based PSO to the RNA secondary structure prediction successfully. For discrete optimization, the authors in [35] introduced the set concept to PSO and applied to data clustering effectively. Recently, the researchers in [36] proposed an updated set-based PSO (S-PSO) for solving discrete optimization problems. The main difference between the newly proposed S-PSO and the previous ones [37, 38, 39] is that the new one incorporated possibilities to velocity set elements for each particle. This update helped to guide the movements of particles towards the optima. S-PSO used traveling salesman problems (TSP) and knapsack problems as examples to demonstrate the good performance of the algorithm. S-PSO is in effect a combination of PSO learning mechanisms and set operations involving probabilistic distributions. Since we can construct the 2D and 3D HP lattice model in a lattice board, the construction path of the folding process could be considered similarly as in TSP. By following the framework of S-PSO, a novel algorithm based on it for solving PFP is proposed in this paper. The algorithm is termed as $DPSO_{HP}$ and it hybridizes the heuristic information and the retrieval strategy in FAC and the possibility concepts in S-PSO to construct the solutions. In the proposed $DPSO_{HP}$ algorithm, a selection strategy using both heuristic information and possibilities is adopted. A position of a particle is defined as a set of elements and a particle's velocity is defined as a set of elements associated with possibilities.

The performance of $DPSO_{HP}$ will be investigated thoroughly and compared with the most competitive CI algorithms for solving PFP. The rest of the paper is organized as follows. Section 2 makes a brief overview of the HP model in PFP and introduces the traditional framework of PSO. In Section 3, some basic concepts and operations for PFP in the S-PSO framework are described. Section 4 presents the implementation of the proposed $DPSO_{HP}$ for solving PFP. Experimental analysis to the characteristics of $DPSO_{HP}$ and its performance compared with other state-of-the-art CI algorithms are discussed in Section 5. Section 6 concludes the paper.

II. Preliminaries Of The Hp Model And Pso

This section first describes the definition of the HP model in PFP, including existed methods for addressing the model. Then the traditional PSO algorithms are introduced.

A. HP Model of PFP

The HP model was first introduced by Dill [2]. In this model, the amino acid sequence of a protein is represented by a sequence of letters consisting of 'H' and 'P', where 'H' stands for the hydrophobic amino acid and the letter 'P' stands for the hydrophilic amino acid. Fig. 1 shows an example of a protein conformation in a 2D HP lattice. The black nodes denote H and the white nodes denote P. The dashed lines represent the H-H bounds. The number 1 indicates the first amino acid in the protein sequence. The conformation is optimal which has minimum free energy of -4. The energy of a protein conformation is defined as the number of adjacent hydrophobic amino acid pairs that are formed in the lattice but are not consecutive in the sequence. Between those two adjacent hydrophobic amino acids there is an H-H bound, which has free energy of -1. In Fig. 1, the conformation has a total free energy of -4 since it has 4 H-H bounds. Given a protein sequence, the objective of the PFP with HP model is to find conformations that have the minimum total free energy. For evolutionary algorithms, the objective function is to minimize the total free energy in HP lattice model. The particles in the proposed algorithm aim to find a conformation path of a protein sequence in the lattice.

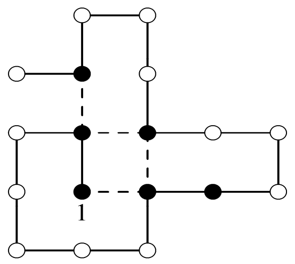


Fig. 1. A conformation of sequence *HHPPPPPHHPPPHPPPHP* in the 2D HP model.

No matter in the 2D or 3D HP model, a protein sequence folds itself to form a self-avoiding path. We use absolute directions in a search space to represent the conformation. For the 2D HP model, the absolute four directions are left (L), right (R), forward (F) and backward (B). In the 3D HP model, there are two more directions up (U) and down (D). A direction starts from one cell and points to one of its adjacent cells. The layout approach is similar to that in a travelling salesman problem, when the cells in the lattice are regarded as cities and the connections between cells are considered as arcs. So we can employ the discrete evolutionary algorithms that perform successfully in TSP to deal with the PFP with HP lattice model.

One of the distinct features of PFP is to construct a self-avoiding path. To avoid overlaps of amino acids in the lattice, the algorithm in [25] used two mechanisms. First, a simple look-ahead mechanism was used to disallow placing any amino acids in a position that can cause stagnation except for the two ending amino acids. Second, if no placement can comply with the look-ahead mechanism, half of the already constructed path was released as retrieval. Different from the method in [25], FAC[4] randomly chose a position and released the corresponding amino acids. It also used a mechanism to ensure that the retrieval in the same direction cannot be performed twice consecutively. In $\mathrm{DPSO_{HP}}$, we follow the self-avoiding folding strategies of FAC[4].

B. Introduction to PSO

PSO[29] is inspired by the interpretations of the movement of organisms in a school of flying birds. The birds are abstracted as particles which search for food. When searching solutions in a continuous space, each particle dynamically adjusts its flying velocities and positions iteratively according to its own experience and other particles' experiences. Learning is a distinguished feature of PSO. Suppose there are m particles searching solutions in an n-dimensional continuous space. Each particle i (i=1,2,…,m) has two attributes: velocity and position, which are denoted as $V_i = <v_i^1, v_i^2, \ldots, v_i^n>$ and $X_i = <x_i^1, x_i^2, \ldots, x_i^n>$, respectively. The basic velocity and position updating rules are defined as follows:

$$v_i^j = v_i^j + c_1 r_i^j (pbest_i^j - x_i^j) + c_2 R_i^j (gbest^j - x_i^j) \quad (1)$$

$$x_i^j = x_i^j + v_i^j \quad (2)$$

where j (j=1,2,…,n) denotes the jth dimension of the particle i; c_1 and c_2 are the acceleration speed constants which weigh the importance of cognitive and social components respectively; r_i^j and R_i^j are randomly generated numbers from [0,1]. The vector $pbest_i = <pbest_i^1, pbest_i^2, \ldots, pbest_i^n>$ is the best solution found by particle i so far and the vector $gbest = <gbest^1, gbest^2, \ldots, gbest^n>$ is the best solutions found by all the particles so far.

After the first PSO was proposed, many researchers developed variants of PSO to improve its performance. The authors in [30] introduced an inertia weight ω to the velocity updating rule as:

$$v_i^j = \omega v_i^j + c_1 r_i^j (pbest_i^j - x_i^j) + c_2 R_i^j (gbest^j - x_i^j) \quad (3)$$

They showed that the inertia weight ω had the effect to balance the global and local search ability of the algorithm.

Another important variant is the comprehensive learning PSO (CLPSO) in [13]. The velocity updating rule is given by:

$$v_i^j = \omega v_i^j + c r_i^j (pbest_{f_i(j)}^j - x_i^j) \quad (4)$$

where $f_i(j)$ defines which particle's experience should be learnt from by particle i for dimension j.

The value of $f_i(j)$ is computed as follows. For each dimension j of particle i, a number is randomly generated from [0,1]. If the number is larger than a parameter Pc called learning probability, then $f_i(j) = i$, which means particle i will learn from its own experience. Otherwise particle i will learn from another particle's experience. Then the algorithm employs a tournament selection strategy to determine the value of $f_i(j)$. Two particles are randomly selected, and the value of $f_i(j)$ is the *ID* of the particle with higher fitness. In this way, all these $pbest_{f_i(j)}^j$ can be derived from different particles' *pbest* positions. To ensure that a particle learns from good directions and saves time on learning from poor directions, CLPSO does not use the learning strategy in every generation. When a particle stops improving for a certain number of generations which is called refreshing gap rg, CLPSO recalculates the value of f_i for the particle. According to the discussions in [36], the velocity updating strategies in CLPSO were very effective.

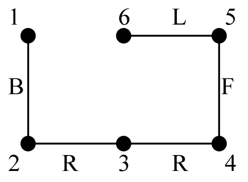
III. Basic Concepts And Operations For PFP In The S-PSO Framework

A. Solution Representation

According to the S-PSO[36] framework, candidate solutions are in effect crisp subsets out of a universal set E of elements. In the 2D HP model for PFP, each element is denoted as an arc (j, d), where j is the amino acid index (j = 1,2,…,n,); n is the number of amino acids and $d \in$ {L,R,F,B} is the folding direction. There are n dimensions $E^1, E^2, \ldots, E^n$ in the PFP with $E^j \subseteq E$ and n is equal to the number of amino acids. Each dimension E^j comprises four arcs that are connected with j, and can be denoted as E^j = {(j,L), (j,R), (j,F), (j,B)}. Each candidate solution X is composed of an element in E^j with j = 1,2,…,n and the arcs must be connected to form a protein folding conformation. X is feasible only when it forms a self-avoiding path.

Fig. 2 shows an example of the representation of the folding conformation. The solid nodes are amino acids in the protein sequence. The solution by particle i is represented as X_i = {(2,B), (3,R), (4,R), (5,F), (6,L)}. Suppose a particle i chooses the first amino acid as the starting point for folding the protein sequence. Then both E^1 and X^1 are empty sets. The amino acids with indices from 2 to 6 have non-empty dimensions and an element is chosen from each dimension to form a self-avoiding path. The representation of the folding conformation in Fig. 2 is described as follows: X_i = {(2,B), (3,R), (4,R), (5,F), (6,L)}.

In the remainder of this paper, we also use *position* to represent a feasible solution found by a particle. The position of a particle i is the same as its solution $X_i = \{X_i^1, X_i^2, \ldots, X_i^n\}$ and the jth dimension of its position is X_i^j.



$E^1 = \Phi$ $X^1 = \Phi$

$E^2 = \{(2,L),(2,R),(2,F),(2,B)\}$ $X^2 = \{(2,B)\}$

$E^3 = \{(3,L),(3,R),(3,F),(3,B)\}$ $X^3 = \{(3,R)\}$

$E^4 = \{(4,L),(4,R),(4,F),(4,B)\}$ $X^4 = \{(4,R)\}$

$E^5 = \{(5,L),(5,R),(5,F),(5,B)\}$ $X^5 = \{(5,F)\}$

$E^6 = \{(6,L),(6,R),(6,F),(6,B)\}$ $X^6 = \{(6,L)\}$

Fig. 2. An example of the representation scheme in S_PSO for the protein folding problem.

B. Folding Velocity

In S-PSO, a velocity is defined as a set of possibilities, which are used to guide the movements of particles towards the optima. Given a universal set E, the velocity V is defined as:

$$V = \{e / p(e) \mid e \in E\} \tag{5}$$

where $p(e) \in [0,1]$ is the possibility for selecting the element e. We use a subscript i to denote the velocity V_i for each particle i, and thus the velocity of dimension j is denoted as $V_i^j, j = 1,2,\ldots,n$.

Fig. 3 illustrates the velocity V_i^2 of particle i for selecting an element in dimension 2 in the PFP. The dashed lines are the possible folding directions. Folding backward has the largest possibility for being chosen by the particle, whereas folding left has the smallest possibility. In the subsequent section, the method for adaptively updating the possibilities in the velocity will be presented.

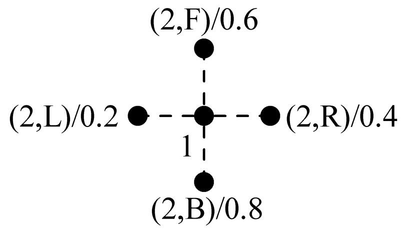


Fig. 3. An example of a velocity.

C. Calculation Operations

Traditional PSOs for continuous optimization involve several arithmetical operations for updating the positions of particles and their velocities. However, for discrete optimization, operations such as additions and subtractions are needed to be redefined. Here we describe the operations for PFP based on the S-PSO framework.

1) Coefficient×Velocity:

Multiply a coefficient by a velocity, the result is also a velocity. Given a coefficient $c(c \geq 0)$ and a velocity $V = \{e/p(e) \mid e \in E\}$ defined on E, the product is:

$$cV = \{e / p'(e) \mid e \in E\} \tag{6}$$

If the multiplication is greater than 1, $p'(e)$ is simply truncated to 1. For example, when $V_i^2 = \{(2,L)/0.2,(2,R)/0.4,(2,F)/0.6,(2,B)/0.8\}$, suppose c=2.0, then we have

$cV_i^2 = \{(2,L)/0.4,(2,R)/0.8,(2,F)/1.0,(2,B)/1.0\}$.

2) Velocity + Velocity:

Given two velocities $V_1 = \{e/p_1(e) \mid e \in E\}$ and $V_2 = \{e/p_2(e) \mid e \in E\}$, V_1 plus V_2 is defined as

$$V_1 + V_2 = \{e / \max(p_1(e), p_2(e)) \mid e \in E\} \tag{7}$$

The possibility after the summation of two velocities equals to the larger possibility of the two velocities. For example, suppose $V_1^2 = \{(2,L)/0.2, (2,R)/0.4, (2,F)/0.6, (2,B)/0.8\}$ and $V_2^2 = \{(2,L)/0.3, (2,R)/0.1, (2,F)/0.6, (2,B)/0.9\}$, then $V_1^2 + V_2^2 = \{(2,L)/0.3, (2,R)/0.4, (2,F)/0.6, (2,B)/0.9\}$.

3) Position- Position:

The minus operator between the positions of two particles is defined the same as the subtraction of two crisp sets, as

$$A - B = \{e \mid e \in A \text{ and } e \notin B\} \tag{8}$$

For example, given $X_i^2 = \{(2,L)\}$ and $pbest_i^2 = \{(2,R)\}$, then $pbest_i^2 - X_i^2 = \{(2,R)\}$. If $X_i^2 = \{(2,R)\}$, then $pbest_i^2 - X_i^2 = \varnothing$.

4) Coefficient×Position:

Multiplying a coefficient by a position is defined as a velocity. Given a coefficient c ($c \geq 0$) and a position A, the product is defined as follows:

$$cA = \{e / p(e) \mid e \in E\} \tag{9}$$

where

$$p(e) = \begin{cases} 1, & \text{if } e \in A \text{ and } c > 1 \\ c, & \text{if } e \in A \text{ and } 0 \leq c \leq 1 \\ 0, & \text{if } e \notin A \end{cases} \tag{10}$$

For example, given $X_i^2 = \{(2,L)\}$ and c = 0.5, multiplying c by X_i^2, then $cX_i^2 = \{(2,L)/0.5\}$, which is a velocity.

IV. Implementation Of The Proposed DPSO$_{HP}$ For Solving Pfp

This section describes the implementation of the proposed DPSO$_{HP}$ in solving PFP in detail. First, we outline the overall framework of the algorithm. Then the velocity updating and position updating processes are described. For simplicity, we only present examples in 2D space for the position updating process. At last, we present the method for extending the algorithm to 3D space.

A. Framework of DPSO$_{HP}$

The overall framework of DPSO$_{HP}$ is presented as Algorithm 1. In the following subsections, we will describe the special velocity updating and position updating processes of DPSO$_{HP}$.

Algorithm 1 DPSO$_{HP}$ algorithm

```
1: Initialization
2: While termination not met
3:     for each particle $i$ ($i$=1,2,…,$m$)
```

```
4:         velocity updating
5:         position updating
6:     end for
7: End While
```

B. Velocity Updating

In S_PSO, the authors showed that the velocity updating strategies in CLPSO obtained the best results in discrete domains. Accordingly, in DPSO$_{HP}$, we use the velocity updating rule in CLPSO with the redefinitions in S_PSO. The arithmetic operators in the velocity updating rule of DPSO$_{HP}$ can be summarized as the following seven steps.

(1) Calculate ωv_i^j by Equation (6);

(2) Calculate the value of $f_i(j)$ by the method mentioned in Section II B;;

(3) Calculate $pbest^j_{f_i(j)} - x^j_i$ by Equation (8);

(4) Generate a random number r_{ij} from [0,1];

(5) Calculate $cr_i^j(pbest^j_{f_i(j)} - x^j_i)$ by Equation (9);

(6) Calculate $\omega v_i^j + cr_i^j(pbest^j_{f_i(j)} - x^j_i)$ by Equation (7);

(7) Updating v_i^j by the result calculated in step (6).

In the construction phase, there are only four directions for a particle to choose. With the effect of inertia weight, the possibility of a direction may decrease very fast. It means that the probability to choose the direction will be very low. As a result, the algorithm can be easily trapped in a local optimum. To address this problem, we define a lowest possibility p_{min}. When a possibility is lower than p_{min}, it will be adjusted to p_{min}.

C. Position Updating

Each particle uses the newly updated velocity to adjust its position in order to improve its solution quality. Since the number of elements in each dimension is limited in the discrete space, a particle needs to reconstruct the path for adjusting the position. Given a protein sequence with length n, i.e. n amino acids, a square lattice board with n+2 rows and n+2 columns is used for the 2D HP model as in Fig.4. Each cell in the lattice is indexed according to their positions on the board from the top left cell 0 to the bottom right cell $(n+2)^2$-1. In this way, when a protein sequence folds to the left, right, forward, or backward, the corresponding increments of the cell index is -1, 1,-(n+2), and n+2. Let $Mid = \lceil n/2 \rceil$, the indices of the two central cells are $Mid\times(n+2)+Mid$ and $Mid\times(n+2)+Mid+1$, and they are termed as the 'left start cell' and the 'right start cell', respectively. The construction of solutions is based on the lattice board. In the following, the path construction process is described, including a path retrieval strategy for fixing infeasible solutions and its extension to 3D HP model.

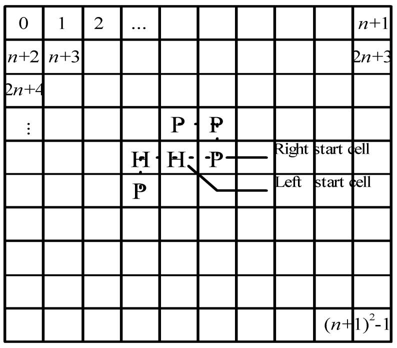


Fig. 4. The square lattice for a protein with n amino acids.

1) Path Construction

The path construction process is similar to the method in [4]. For a protein sequence, the indices of the two middle amino acids are Mid-1 and Mid, respectively. At the beginning of the construction phase, each particle chooses the two middle amino acids and places them in the two central cells in the lattice board. After that, the particles randomly choose to fold the left part or right part of the protein sequence. Fig. 4 gives part of the folding conformation of a protein in the square lattice after several construction steps. The numbers in the figure indicate the indices of the cells.

During the construction process, heuristic information is used to bias the search. Because the mission for PFP is to find the conformation with the minimum energy, a direction that can result more H-H bounds will have a higher probability to be chosen to fold the protein. For hydrophobic amino acids, the heuristic values are calculated as:

$$\eta_{jd} = h_{jd} + 1 \tag{11}$$

where h_{jd} is the number of new H-H bounds obtained by placing s_j with direction d. For example, in 2D HP model, h_{jd} could be 0, 1, 2, or 3. In order to ensure that η_{jd} is greater than 0, we plus 1 to h_{jd}. This is important to not exclude any directions in the construction phase by making sure that the probability calculated in equation (13) is greater than 0.

For polar amino acids, the heuristic values are influenced by the density of grids and the number of new P-P bounds as:

$$\eta_{jd} = v_{jd} + h'_{jd} + 1 \tag{12}$$

where v_{jd} is the number of directions that the next amino acid can choose after placing s_j and h'_{jd} is the number of new P-P bounds obtained. Polar amino acids tend to surround hydrophobic amino acids and contact water molecules directly. Therefore, it is better to fold them to the grids that involve more vacant grids. Although polar amino acids contribute no new H-H bounds, forming new P-P bounds can increase the chances for the formation of other new H-H bounds.

To choose the direction in each construction step, the proportional selection method is used. Each direction d for folding an amino acid s_j is determined both by the heuristic information and the possibilities of the elements as calculated as following:

$$pro_{jd} = \frac{p_{jd} \times \eta_{jd}^{\beta}}{\sum_{q \in allowed} p_{jq} \times \eta_{jq}^{\beta}} \qquad (13)$$

where β is a parameter for controlling the relative importance of the heuristic information. p_{jd} is the pheromones that are released on the directed virtual connections between adjacent squares, where i=0,1,2,..,(n+2)^2-1 and d=L, R, F, B.

2) *Path Retrieval*

During the path construction, an amino acid cannot be placed if all possible locations are already occupied by other amino acids. In that case, the protein cannot be folded any more during the construction phase and it is termed *stagnation*. Therefore, a path retrieval strategy is used to adjust an infeasible solution into a feasible one.

A position is randomly been chosen and the corresponding amino acids are released when stagnation happens. A mechanism is also used to ensure that the retrieval in the same direction is not performed twice consecutively. The retrieval strategy employed in DPSO$_{HP}$ is described as follows. Given a sub-sequence $\{s_{left},..., s_{startL}, s_{startR},..., s_{right}\}$, *left* and *right* are the indices of the left most and right most amino acids that have already been constructed; *startL* and *startR* are the indices of the two middle amino acids. If the stagnation happens on the left path of the protein sequence, the procedure of left path retrieval is performed, whereas the procedure of right path retrieval is performed for the right stagnation. Fig. 5 illustrates the flowchart for performing path retrieval to relieve stagnation.

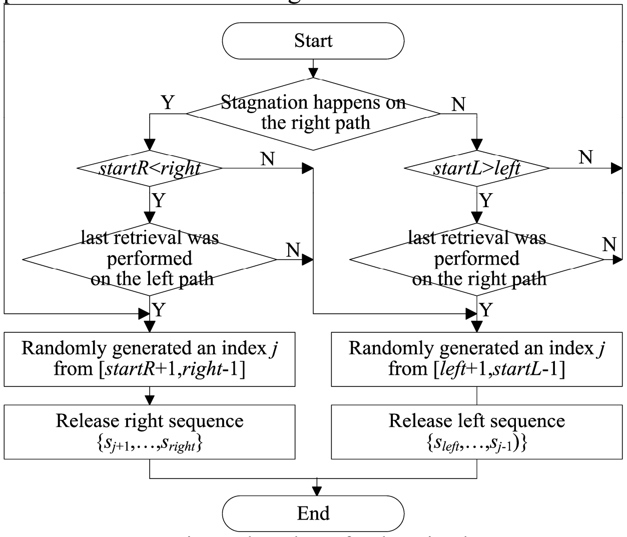

Start
Stagnation happens on the right path
Y
N
startR<right
N
Y
startL>left
N
Y
last retrieval was performed on the left path
N
Y
last retrieval was performed on the right path
N
Y
Randomly generated an index j from [startR+1,right-1]
Randomly generated an index j from [left+1,startL-1]
Release right sequence {s_{j+1},…,s_{right}}
Release left sequence {s_{left},…,s_{j-1})}
End

Fig. 5. Flow chart of path retrieval.

For example, when the stagnation happens on the right path currently, if *startR<right* and the last retrieval is not performed on the right path, the right part of the sequence is released (the left part of Fig. 5). To retrieve the right part of the sequence, an index *j* is generated randomly from [*startR*+1, *right*-1] and then the sub-sequence $\{s_{j+1},..., s_{right}\}$ is released. However, if $startR \geq right$ or the last retrieval is performed on the right path, the left part of the sequence is released. An index *j* is generated randomly from [*left*+1, *startL*-1] and then the sub-sequence $\{s_{left},...,s_{j-1}\}$ is released.

Fig. 6 shows an example for the retrieval where the black nodes denote the two middle amino acids. For folding a No. 3 amino acid, the sequence stagnates on the left path (Fig. 6a), so the left path retrieval is performed. Suppose a random integer *j*=6 is generated, then the amino acids from indices 4 to 5 are released. Fig. 6b shows the path after the retrieval. It can be seen that the stagnation is not cleared because the left stagnation will happen again after several construction steps. Fig. 6c shows the conformation when the left stagnation happens again. Since the left path retrieval has been performed before, the right path retrieval is performed this time. Suppose the randomly generated integer *j*=13, then amino acids from indices 14 to 20 are released. The path after the retrieval is shown in Fig. 6d and it can be seen that the stagnation is now cleared.

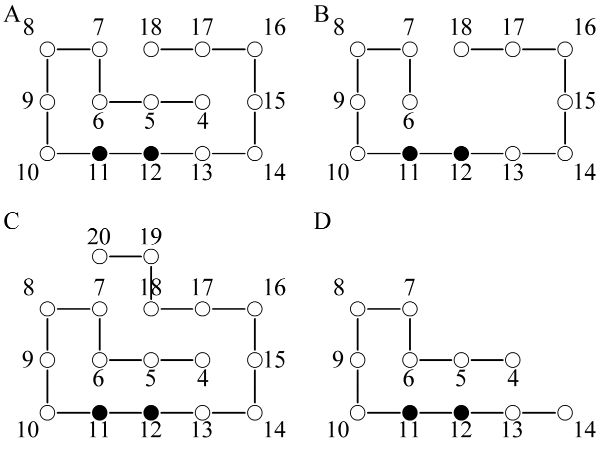


Fig. 6. An example of path retrieval.

3) *Further Extension to the 3D model*

To extend the algorithm to the 3D HP model, some changes are listed as follows. First, there are two more absolute directions up (U) and down (D) in the 3D model. The lattice board is changed to be a cubic lattice with the size of $(n+2)^3$. The cells in the lattice are indexed from 0 to $((n+2)^3-1)$. When a particle chooses to go up or down, the corresponding increments of the cell index is $-(n+2)^2$ and $(n+2)^2$. The indices of the two central cells are $Mid\times[(n+2)^2+n+3]$ and $Mid\times[(n+2)^2+n+3]+1$ respectively.

V. Experimental Analysis

In this section, we first investigate the influence of different parameter settings for the performance of DPSO$_{HP}$. The performances of DPSO$_{HP}$ in solving 2D HP PFP and 3D HP PFP are then analyzed and compared with some state-of-the-art computational intelligence algorithms such as IA, EMC, GA, and EDAs.

Table 1 tabulates the benchmark HP protein sequences used in this paper, where *l* is the number of amino acids in the protein sequence and E^* is the best-known free energy of the sequence. The default parameter settings of the experiments are set as c=2.0, ω=0.95, β=3, p_{min}=0.05, m=50. We follow the CLPSO [29] to set the refreshing gap r_g=7 and:

$$Pc_i = 0.05 + 0.45 \times \frac{\exp(\frac{10(i-1)}{m-1}) - 1}{\exp(10) - 1} \quad (14)$$

where i is the index of the particle and m is the swarm size. DPSO$_{HP}$ was implemented in C++ language on a PC with 2.0GHz Intel Pentium E2180 CPU. The experimental results of all the other compared algorithms were obtained from the reference papers. We set the same evaluation numbers as the compared algorithms did and test DPSO$_{HP}$ on those protein sequences which were tested in the compared algorithms.

TABLE 1.
BENCHMARKS OF PROTEIN SEQUENCES

No.	*l*	*E**	protein sequence
1	20	-9	$HPHP_2H_2PHP_2HPH_2P_2HPH$
2	24	-9	$H_2P_2(HP_2)_6H_2$
3	25	-8	$P_2HP_2(H_2P_4)_3H_2$
4	36	-14	$P_3H_2P_2H_2P_5H_7P_2H_2P_4H_2P_2HP_2$
5	48	-23	$P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$
6	50	-21	$H_2(PH)_3PH_4P(HP_3)_2HP_4(HP_3)_2HPH_4(PH)_3PH_2$
7	60	-36	$P_2H_3PH_8P_3H_{10}PHP_3H_{12}P_4H_6PH_2PHP$
8	64	-42	$H_{12}(PH)_2((P_2H_2)_2P_2H)_3(PH)_2H_{11}$
9	85	-53	$H_4P_4H_{12}P_6(H_{12}P_3)_3HP_2(H_2P_2)_2HPH$
10	100	-48	$P_6HPH_2P_5H_3PH_5PH_2P_4H_2P_2H_2PH_5PH_{10}PH_2PH_7P_{11}H_7P_2HPH_3P_6HPH_2$
11	100	-50	$P_3H_2P_2H_4P_2H_3(PH_2)_2PH_4P_8H_6P_2H_6P_9HPH_2PH_{11}P_2H_3PH_2PHP_2HPH_3P_6H_3$

A. Investigations of Parameter Settings in DPSO$_{HP}$

The influence of the heuristic reinforcement value β and the inertia weight ω in DPSO$_{HP}$ is analyzed by the average run-time. Each group of parameters is tested for 30 independent trials. The average run-time required for reaching the optimal solution is recorded for each trial. Fig. 7 illustrates the influence of different parameter settings β and ω for the sequences 1, 2, and 3. Panels from row 1 to row 3 illustrate influences of sequences 1-3. Panels on the left illustrate influences for β, and panels on the right illustrate influences for ω. The other parameter values are fixed. The characteristics of the two parameters to the performance can be concluded as:

(1) When the heuristic reinforcement value β increases, the time needed to reach the optima becomes shorter for sequences 1 and 2. For sequence 3, the best result is obtained when β=3.

(2) When the inertia weigh ω is about 0.95, the performance of the algorithm is good in most of the test cases. Overall, the influence of ω is not so important as β.

According to the above experimental analysis, we set β=3 and ω=0.95 in all of our experiments.

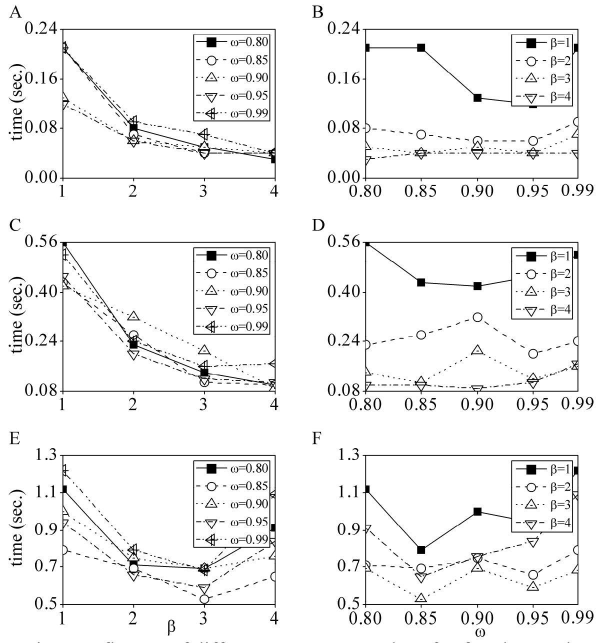


Fig. 7. Influence of different parameters values for β and ω on the average run-time.

B. Experiments in the 2D HP Model

In the first experiments we compare DPSO$_{HP}$ with IA [3, 25], EMC [13], GA [21], and some EDAs [5] in solving PFP with 2D HP model. Sequences 1 to 8 in Table 1 are used as the test instances for comparing DPSO$_{HP}$ with IA, EMC, and GA, where sequences 1 to 11 are used for comparing with the EDAs in [5].

1) Comparison with IA, EMC and GA

The parameter settings of EMC and GA can be referred to [13, 21]. IA used memory B cells, with the aging values τ_B=1 and τ_{Bmem}=5. All the algorithms were tested for 30 independent trials with a maximum number of energy evaluations 10^7. The results are tabulated in Table 2 to Table 4. The best results are represented in bold in the tables.

Table 2 compares the average number of energy evaluations and success rates of IA and DPSO$_{HP}$. *AE* indicates the average number of energy evaluations and *SR* indicates the percentage of times which the optimal solution is found in all trials. For short sequences 1-4 (l≤36), DPSO$_{HP}$ and IA both find the optimal solutions in all trials successfully. However, the average number of energy evaluations used by DPSO$_{HP}$ is much smaller than those in IA for all short sequences. For sequences 5, 7 and 8, DPSO$_{HP}$ can find the optimal solutions with higher success rates than IA can. Especially for sequences 7 and 8, IA cannot find the optimal solutions in any one trial, whereas DPSO$_{HP}$ obtains the success rate 43.33% and 40% respectively. For sequence 6, IA with a success rate 100% performs better than DPSO$_{HP}$ with a success rate of 90%.

TABLE 2.
COMPARISONS OF AVERAGE NUMBER OF ENERGY EVALUATIONS AND SUCCESS RATE.

No.	DPSO$_{HP}$		IA	
	AE	*SR*	*AE*	*SR*
1	**2174**	**100**	23710	**100**
2	**6525**	**100**	69816.7	**100**
3	**39173**	**100**	269513.9	**100**
4	**46007**	**100**	2032504	**100**
5	**1044558**	**100**	6403985.3	56.67
6	3088689	90	**778906.4**	**100**
7	**4009735**	**43.33**	-	0
8	**4995706**	**40**	-	0

- indicates that the algorithm can't obtain the optimal solution in all trials.

Table 3 compares the best energy values obtained by EMC, IA, GA, and DPSO$_{HP}$. DPSO$_{HP}$ can find the optimal solutions for all the eight sequences, whereas the other algorithms cannot find the optimal solutions for sequences 7 and 8.

Table 4 compares the number of energy evaluations required for the best trial to achieve the optima by DPSO$_{HP}$, IA, EMC and GA. DPSO$_{HP}$ obtains the optimal results with the lowest number of energy evaluations except for sequence 6.

2) *Comparison with EDAs*

Because three more sequences number 9 to 11 with longer lengths are used in this comparative experiment, a larger number of energy evaluations are conducted. The EDAs in [5] use a population size of m=5000 individuals, a maximum of g=5000 generations, and the truncation selection T=0.1. Since the EDAs use the best elitism, the maximum number of energy evaluations can be calculated as m+m(1-T)(g-1), which is slightly larger than 2×10^7. Hence, we run DPSO$_{HP}$ with a maximum number of energy evaluations 2×10^7 in comparisons with the EDAs in [5]. All the algorithms are tested for 50 independent trials.

TABLE 3.
COMPARISONS OF BEST ENERGY VALUES

No.	*E**	DPSO$_{HP}$	IA	EMC	GA
1	-9	**-9**	**-9**	**-9**	**-9**
2	-9	**-9**	**-9**	**-9**	**-9**
3	-8	**-8**	**-8**	**-8**	**-8**
4	-14	**-14**	**-14**	**-14**	**-14**
5	-23	**-23**	**-23**	**-23**	**-23**
6	-21	**-21**	**-21**	**-21**	**-21**
7	-36	**-36**	-35	-34	-35
8	-42	**-42**	-39	-37	-39

TABLE 4.
COMPARISONS OF BEST NUMBER OF ENERGY EVALUATIONS

No.	DPSO$_{HP}$	IA	EMC	GA
1	**217**	1925	9374	30492
2	**350**	2479	6929	30491
3	**1780**	4212	7202	20400
4	**2754**	43416	12447	301339
5	**21486**	37269	165791	126547
6	153984	**18919**	74613	592887

Table 5 compares the performance of DPSO$_{HP}$ and EDAs with *BV* indicating the best energy value obtained in the test trials. For sequences 1 and 2, all algorithms can find the optimal solutions in all trials successfully. For sequences 3, only DPSO$_{HP}$ and MK-EDA can find the optimal solution in all trials successfully. For sequences 4-6, DPSO$_{HP}$ can find the optimal solutions in all trials successfully, whereas the EDAs fail in some trials. Especially for sequences 4 and 5, EDAs can only find the optimal solutions in less than 20% of all trails. Note that DPSO$_{HP}$ can find the optimum -53 of sequence 9, compared to MK-EDA which can only find the best energy value of -52. For the longest sequences 10 and 11 with the optima -48 and -50 respectively, DPSO$_{HP}$ and MT-EDA can both find the near optima -47 and -48, whereas DPSO$_{HP}$ has higher success rates to find the near optima.

C. *Experiments in the 3D HP Model*

In these experiments, the performances of DPSO$_{HP}$ in solving PFP with 3D HP lattice model are analyzed. DPSO$_{HP}$ is compared with a hybrid GA [21], IA [3, 22], and EDAs [5]. The hybrid GA uses an absolute encoding and a repair-based approach. Sequences 1-8 in Table 1 are used as the test instances for the 3D model. All the algorithms are tested for 50 independent trials with a maximum number of energy evaluations 10^5. To our best knowledge, the optimum energy values for the tested sequences in 3D HP lattice model are yet known. We consider that the less energy values an algorithm obtained, the better the algorithm is. The best values, means, and standard deviations (σ) are shown in Table 6. The best results among the compared algorithms are marked in bold.

TABLE 5.
RESULTS OF DPSO$_{HP}$ AND EDAS IN 2D HP MODEL

No.	*E**	DPSO$_{HP}$		MK-EDA		Tree-EDA		MT-EDA	
		BV	*S*	*BV*	*S*	*BV*	*S*	*BV*	*S*
1	-9	**-9**	**100**	**-9**	**100**	**-9**	**100**	**-9**	**100**
2	-9	**-9**	**100**	**-9**	**100**	**-9**	**100**	**-9**	**100**
3	-8	**-8**	**100**	**-8**	**100**	-8	88	-8	90
4	-14	**-14**	**100**	-14	8	-14	4	-14	16
5	-23	**-23**	**100**	-23	14	-23	18	-23	4
6	-21	**-21**	**100**	-21	86	-21	98	-21	96
7	-36	**-36**	**50**	-35	10	-35	12	-35	18
8	-42	**-42**	**60**	-42	6	-41	10	-42	12
9	-53	**-53**	2	-52	4	-51	2	-50	4
10	-48	**-47**	**8**	-46	6	-46	16	-47	2
11	-50	**-48**	**4**	-47	2	-47	12	-48	2

TABLE 6.
RESULTS OF DPSO$_{HP}$, HYBRID GA, IA AND TREE-EDA IN 3D HP MODEL

No.	DPSO$_{HP}$		hybrid GA		IA		Tree-EDA	
	BV	*mean±σ*	*BV*	*mean±σ*	*BV*	*mean±σ*	*BV*	*mean±σ*
1	**-11**	**-11.00±0.00**	-11	-9.84±0.86	-11	-10.90±0.32	-11	-11.00±0.00
2	**-13**	**-13.00±0.00**	-11	-10.00±0.87	-13	-12.22±0.65	-13	-12.86±0.16
3	**-9**	**-9.00±0.00**	-9	-8.64±0.69	-9	-8.88±0.48	-9	-8.90±0.09
4	**-18**	**-18.00±0.00**	-18	-13.72±1.41	-18	-16.08±1.02	-18	-16.34±0.51

5	**-31**	**-28.50±0.91**	-28	-18.90±2.08	-28	-24.82±0.71	-27	-23.62±1.83
6	-30	-24.78±1.11	-22	-19.06±1.46	-23	-22.08±1.43	**-30**	**-26.00±2.82**
7	**-51**	**-48.94±0.87**	-38	-32.28±3.09	-41	-39.02±0.50	-37	-32.94±1.53
8	**-54**	**-48.14±2.05**	-36	-30.84±2.55	-42	-39.07±1.20	-44	-34.70±6.87

It can be seen in Table 6 that not only the best energy values but also the mean energy values reached by DPSO$_{HP}$ are the best in all algorithms except for the sequence 6. For sequences 1 to 4, DPSO$_{HP}$ obviously outperforms the other algorithms since it can find the optimal energy values in all trails. It is remarkable that DPSO$_{HP}$ achieves lower energy values in sequences 5, 7, and 8. In these sequences, the best energy values obtained by the other algorithms are -28, -41, and -44 respectively, whereas DPSO$_{HP}$ obtains the much better results of -31, -51, and -54. For sequence 6, DPSO$_{HP}$ can still obtain the same best value as that by Tree-EDA but with a slightly poorer mean energy value.

Since the algorithm cannot obtain satisfying results within the predefined maximum number of energy evaluations for the longer protein sequences, we use a larger number of energy evaluations to investigate whether the performance of the algorithm can be better. The maximum number of energy evaluations 5×10^6 is used by DPSO$_{HP}$ and the results are compared with those of MK-EDA, Tree-EDA, and MT-EDA with the same maximum number of energy evaluations. The results are shown in Table 7. We can see that by using more energy evaluations, DPSO$_{HP}$ can find lower energy values for sequences 5-8. DPSO$_{HP}$ is more stable for obtaining high-quality solutions than the compared EDAs. It can be seen that not only the best energy values, but also the mean energy values and the standard variances of DPSO$_{HP}$ become better and outperform the EDAs.

TABLE 7.
RESULTS OF DPSO$_{HP}$ AND EDAS IN THE 3D HP MODEL

No.	DPSO$_{HP}$		MK-EDA		Tree-EDA		MT-EDA	
	BV	*mean±σ*	*BV*	*mean±σ*	*BV*	*mean±σ*	*BV*	*mean±σ*
1	**-11**	**-11.00±0.00**	-11	-10.82±0.38	-11	-10.68±0.51	-11	-10.84±0.37
2	**-13**	**-13.00±0.00**	-13	-12.02±0.94	-13	-11.30±0.85	-13	-11.88±0.93
3	**-9**	**-9.00±0.00**	-9	-8.96±0.19	-9	-8.92±0.27	**-9**	**-9.00±0.00**
4	**-18**	**-18.00±0.00**	-18	-16.40±0.80	-18	-16.24±0.83	-18	-16.50±0.96
5	**-31**	**-30.97±0.18**	-29	-27.24±0.92	-29	-26.88±0.93	-29	-27.06±1.08
6	**-31**	**-29.43±0.63**	-29	-25.70±1.26	-31	-25.94±1.58	-28	-25.74±1.22
7	**-53**	**-52.23±0.57**	-49	-46.30±2.04	-49	-43.78±3.10	-48	-42.00±6.76
8	**-59**	**-55.07±1.26**	-52	-46.78±2.28	-49	-43.72±2.43	-50	-45.64±2.03

VI. CONCLUSION

This paper proposes a novel DPSO$_{HP}$ algorithm for solving PFP with 2D and 3D HP lattice models. The algorithm adopts the framework of S-PSO for solving discrete optimization problems. In DPSO$_{HP}$, the PSO learning mechanisms and the set operations are incorporated to search for the optimal protein conformation of a protein sequence. Particles start from the center of the lattice board and construct the protein conformation from the middle of the protein sequence. A selection strategy using both heuristic information and possibilities is used in the algorithm. In this paper, the overall performance of DPSO$_{HP}$ has been analyzed in terms of solution quality and stability. The experimental results show that the proposed DPSO$_{HP}$ is promising and more effective than the other state-of-the-art evolutionary algorithms for solving the PFP with 2D and 3D HP lattice models.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China (NSFC) projects with No. 61202296 and 61202130, the National High-Technology Research and Development Program ("863" Program) of China under Grand No. 2013AA01A212, the Natural Science Foundation of Guangdong Province project with No. S2012030006242 and the State Key Laboratory of Software Engineering project with No. SKLSE2012-09-08.

REFERENCES

[1] C. B. Anfinsen, E. Haber, M. Sela and F. H. White, "The kinetics of the formation of native ribonuculease during oxidation of the reduced polypetide chain," Proc Natl Acad Sci, 47, pp.1309–1314. 1961.

[2] K. A. Dill, "Theory for the folding and stability of globular proteins," *Biochemistry*, 24, pp.1501–1512. 1985.

[3] V. Cutello and G. Nicosia, "An immune algorithm for protein structure prediction on lattice models," *IEEE T Evolut Comput*, vol.11, no.1, pp.101–117, 2007.

[4] X. M. Hu, J. Zhang, J. Xiao and Y. Li, "Protein folding in hydrophobic-polar lattice model: a flexible ant colony optimization approach," *Protein Peptide Lett*, 15, pp.469–477, 2008.

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

[6] F. H. Stillinger, T. H. Gordon, and C. L. Hirshfeld, "Toy model for protein folding," *Phys Rev E*, vol 48, no 2, pp.1469–1477, 1993.

[7] J. D. Hirst, "The evolutionary landscape of functional model proteins," *Protein Eng*, 12, pp.721–726, 1999.

[8] M. S. Li, D. K. Klimov and D. Thirumalai, "Folding in lattice models with side chains," *Comput Phys Commun*, vol 147, no 1, pp.625–628, 2002.

[9] B. Chen, L. Li and J. Lu, "A novel EDAs based method for HP model protein folding," *CEC'09,* pp.309–315, 2009.

[10] H. Lu and G. Yang, "Extremal Optimization for protein folding simulations on the lattice," *Comput Math Appl*, 57, pp.1855–1861, 2009.

[11] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, "On the complexity of protein folding," *J Comput Biol*, vol 5, no 3, pp.423–466, 2008.

[12] N. Krasnogor, W. E. Hart, J. Smith and D. A. Pelta, "Protein structure prediction with evolutionary algorithms," *GECCO*, pp.1596–1601, 1999.

[13] F. Liang and W. H. Wong, "Evolutionary Monte Carlo for protein folding simulations," *J Chem Phys*, vol.115, no. 7, pp.3374–3380, 2001.

[14] R. Ramakrishnan, J. F. Pekny, and B. Ramachandran, "A dynamic Monte Carlo algorithm for exploration of dense conformational spaces in heteropolymers," *J Chem Phys*, vol. 106, no.6, pp.2418–2424, 1997.

[15] JF. Liu, G. Li and J. Yu. "Protein-folding simulation of the

hydrophobic-hydrophilic model by combining pull moves with energy landscape paving", *Phys Rev E*, 84(3), 2011.

[16] C. Cotta, "Protein structure prediction using evolutionary algorithms hybridized with backtracking," *Lect Notes Comput Sc*, 2687, pp.321–328, 2003.

[17] M. T. Hoque, M. Chetty and L. S. Dooley, "Non-isomorphic coding in lattice model and its impact for protein folding prediction using genetic algorithm," *IEEE CIBCB*, pp.1–8, 2006.

[18] T. Z. Jiang, Q. H. Cui, G. H. Shi and S. D. Ma, "Protein folding simulations of the hydrophobic-hydrophilic model by combining tabu search with genetic algorithms," *J Chem Phys*, vol. 119, no. 8, pp.4592–4596, 2003.

[19] M. V. Judy and K. S. Ravichandran, "A solution to protein folding problem using a genetic algorithm with modified keep best reproduction strategy," *CEC'07*, pp.4776–4780, 2007.

[20] R. König and T. Dandekar, "Improving genetic algorithms for protein folding simulations by systematic crossover," *BioSystems*, 50, pp.17–25, 1999.

[21] R. Unger and J. Moult, "Genetic algorithms for protein folding simulation," *J Mol Biol*, 231, pp.75-81, 1993.

[22] V. Cutello, G. Nicosia and M. Pavone, "An immune algorithm with hyper-macromutations for the Dill's 2D hydrophobic-hydrophilic model," *CEC'04*, pp.1074–1080, 2004.

[23] C. P. Almeida, R. A. Goncalves, M. C. Goldbarg, E. F. G. Goldbarg and M. R. Delgado, "TA-PFP: A transgenetic algorithm to solve the protein folding problem," *in Proceedings of the 7th IEEE International conference on Intelligent System Decision and Application*, pp.163–168, 2007.

[24] R. Bitello and H. S. Lopes, "A differential evolution approach for protein folding," *IEEE CIBCB*, pp.1–5, 2006.

[25] A. Shmygelska, R. A. Hernāndez and H. H. Hoos, "An ant colony optimization algorithm for the 2D HP protein folding problem," *Lect Notes Comput Sc*, 2463, pp.40–52, 2002.

[26] A. Shmygelska and H. H. Hoos, "An improved ant colony optimisation algorithm for the 2D HP protein folding problem," *Lect Notes Artif Intell*, 2671, pp.400–417, 2003.

[27] A. Shmygelska and H. H. Hoos, "An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem," *BMC Bioinformatics*, vol.6, no.30, pp.1–22, 2005.

[28] R. Santana, P. Larrañaga and J. A. Lozano, "Protein folding in 2-dimensional lattices with estimation of distribution algorithms," *Lect Notes Comput Sc*, 3337, pp.388–398, 2004.

[29] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *in Proceedings of the IEEE International Conference on Neural Networks*, pp.1942–1948, 1995.

[30] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE T Evolut Comput*, vol.10, no.3, pp.281–295, 2006.

[31] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *in Proceedings of the IEEE International Conference on Evolutionary Computation*, pp.69–73, 1998.

[32] Z. H. Zhan, J. Zhang, Y. Li and S. H. Chung, "Adaptive particle swarm optimization," *IEEE T Syst Man Cy B*, vol.39, no.6, pp.1362-1381, 2009.

[33] A. Bãutu and H. Luchian. "Protein structure prediction in lattice models with particle swarm optimization", in *proceedings of the 7th international conference ANTS 2010*, pp. 512-519.

[34] M. Neethling and A. P. Engelbrecht, "Determining RNA secondary structure using set-based particle swarm optimization," *in Proceedings of the IEEE International Conference on Evolutionary Computation*, pp.1670-1677, 2006.

[35] C. B. Veenhuis, "A set-based particle swarm optimization method," *Parallel Problem Solving from Nature - PPSN X, Lect Notes Comput Sc*, vol. 5199, pp.971-980, 2008.

[36] W. N. Chen, J. Zhang, S. H. Chung, W. L. Zhong, W. G. Wu and Y. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE T Evolut Comput*, vol.14, no.2, pp.278-300, 2010.

[37] H. Zheng, M. Hou and Y. Wang, "An efficient hybrid clustering-PSO algorithm for anomaly intrusion detection," *J of Software*, vol. 6, no. 12, pp. 2350-2360, 2011.

[38] L. Shu and L. Yang, "A modified PSO to optimize manufacturers production and delivery," *J of Software*, vol. 7, No. 10, pp. 2325-2332, 2012.

[39] D. Pan, Y. Ci, M. He and H. He, "An improved quantum-behaved particle swarm optimization algorithm based on random weight," *J of Software*, vol. 8, no. 6, pp. 1327-1332, 2013.

Jing Xiao received the B.S and M.S degrees in computer science from Wuhan University, Wuhan, China, in 1997 and 2000, respectively, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005. Now she is an associate professor in the School of Computer Science, South China Normal University, Guangzhou, China. Previously she was with the Department of Computer Science, Sun Yat-sen University. Her current research interests include evolutionary computation and text/bio-information mining.

Liang-Ping Li received the B. S degree in computer science from the South China University of Science and Technology in 2009 and the M. S degree from the Sun Yat-sen University in computer science in 2012. His research interest is evolutionary computation.

Xiao-Min Hu received the bachelor's degree in computer science and the PhD degree in computer science from the Sun Yat-sen University, Guangzhou, China, in 2006 and 2011, respectively. She is currently a lecturer with the School of Public Health, Sun Yat-sen University, China. Her research interests include evolutionary computation and its applications on bioinformatics.