Computational Biology and Chemistry 54 (2015) 1–12

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/compbiolchem

A balance-evolution artificial bee colony algorithm for protein structure optimization based on a three-dimensional AB off-lattice model

Bai Li[a,b], Raymond Chiong[c,*], Mu Lin[d]

[a] School of Control Science and Engineering, Zhejiang University, Hangzhou 310027, PR China
[b] School of Advanced Engineering, Beihang University, Beijing 100191, PR China
[c] School of Design, Communication and Information Technology, The University of Newcastle, Callaghan, NSW 2308, Australia
[d] College of Biomedical Engineering & Instrument Science, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 12 September 2014
Received in revised form 16 November 2014
Accepted 19 November 2014
Available online 22 November 2014

Keywords:
Protein structure optimization
Amino-acid sequences
AB off-lattice model
Balance-evolution artificial bee colony algorithm

ABSTRACT

Protein structure prediction is a fundamental issue in the field of computational molecular biology. In this paper, the AB off-lattice model is adopted to transform the original protein structure prediction scheme into a numerical optimization problem. We present a balance-evolution artificial bee colony (BE-ABC) algorithm to address the problem, with the aim of finding the structure for a given protein sequence with the minimal free-energy value. This is achieved through the use of convergence information during the optimization process to adaptively manipulate the search intensity. Besides that, an overall degradation procedure is introduced as part of the BE-ABC algorithm to prevent premature convergence. Comprehensive simulation experiments based on the well-known artificial Fibonacci sequence set and several real sequences from the database of Protein Data Bank have been carried out to compare the performance of BE-ABC against other algorithms. Our numerical results show that the BE-ABC algorithm is able to outperform many state-of-the-art approaches and can be effectively employed for protein structure optimization.

1. Introduction

Proteins can be regarded as the primary building blocks in all living organisms (Söding and Lupas, 2003). It is widely acknowledged that the functions of proteins are closely associated with their structures (Freitas et al., 2010), and therefore, understanding the structure of proteins is an important research area in biological sciences (Joshi and Jyothi, 2003; May et al., 2014). In early 2013, there had been approximately 81,700 protein structures deposited in the Protein Data Bank (PDB) database (Bagaria et al., 2013). Most of these protein structures were determined by X-ray diffraction and nuclear magnetic resonance spectroscopy.

Due to strict laboratory requirements and heavy operation burdens, however, there is a huge gap between the number of known amino acid sequences and the deposited structures (Venkatesan et al., 2013; Sousa et al., 2006). It was reported that the number of experimentally determined structures is two orders of magnitude behind the number of protein structures (Poole, 2011). Consequently, many researchers in computational biology have focused their interests on predicting protein structures from the given amino acid sequences (Dorn et al., 2014; Kim et al., 2005). Protein structure prediction refers to the prediction of a three-dimensional protein structure through its primary structure information (Zhang et al., 2010). Here, a three-dimensional configuration is constituted through arranging a sequence of basic structure elements (i.e., α-helix, β-strand and coil).

The thermodynamic hypothesis, originally proposed by Anfinsen (1973), states that native structures of proteins correspond to free-energy minima. Although Anfinsen's pioneering work has enabled researchers to pursue the native structures with minimal free-energy values, solving such a problem is still too difficult for realistic protein models (Kim et al., 2005). Therefore, a critical problem in this area is how many trivial details in protein folding mechanisms can be neglected to establish simplified models (Bachmann et al., 2005). The HP lattice model (Dill, 1985) is one such simplified model, in which all the amino acids in a protein are classified into two categories, namely the hydrophobic and the hydrophilic. The HP lattice model requires all the amino acids to be located inside cubic lattices (Huang et al., 2010). An off-lattice

* Corresponding author. Tel.: +61 2 4921 7367.
E-mail addresses: libai@zju.edu.cn (B. Li), Raymond.Chiong@newcastle.edu.au (R. Chiong), linmudl@zju.edu.cn (M. Lin).

http://dx.doi.org/10.1016/j.compbiolchem.2014.11.004
1476-9271/© 2014 Elsevier Ltd. All rights reserved.

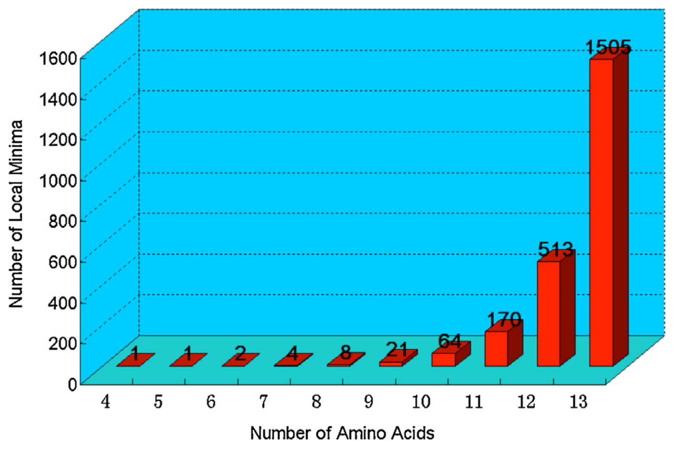

Fig. 1. The growth of the number of local minima vs. the increase in the protein sequence length (redrawn from Rossi and Ferrando, 2009).

generalization of the HP lattice model, namely the AB off-lattice model, was proposed by Stillinger et al. (1993), where all the "binarized" amino acids are linked up by unbendable but free-to-rotate chemical bonds. The AB off-lattice model was first applied in two dimensions and then extended to three dimensions, partially with modifications taking implicitly into account additional torsional energy contributions of each bond (Bachmann et al., 2005).

Considering its advantages in terms of flexibility and accuracy when presenting the underlying locations of amino acids, we adopt the three-dimensional AB off-lattice model to describe the mechanisms of protein folding in a relatively precise but nontrivial manner. In doing so, the original protein structure prediction scheme is transformed into a numerical optimization problem. Such an optimization problem has been confirmed to be NP-hard (Unger and Moult, 1993; Hart and Istrail, 1997; Pierce and Winfree, 2002). As an example, Fig. 1 shows the increase in the number of local minima versus the protein sequence length (Rossi and Ferrando, 2009). Some previous research studies have shown that the number of local minima would increase exponentially with the number of amino acids in a protein sequence (Stillinger and Weber, 1984; Stillinger, 1999). This gives rise to a large number of intelligent or heuristic computational methods for addressing the problem (Li et al., 2013, 2014; Hsu et al., 2003; Liang, 2004; Chen and Huang, 2006; Kim et al., 2007; Zhang and Cheng, 2008; Zhu et al., 2009; Liu et al., 2009, 2013; Chen et al., 2011; Kalegari and Lopes, 2013).

The artificial bee colony (ABC) algorithm is one such method. It is a swarm intelligence algorithm inspired by the foraging behavior of bee swarms (Karaboga and Basturk, 2007). It contains a two-phase searching framework, i.e., local exploitation and global exploration, during the optimization process. Various studies using different numerical benchmark tests have confirmed that the ABC algorithm possesses competitive advantages compared to other well-known evolutionary computation methods (Karaboga and Akay, 2009; Krishnanand et al., 2009; Li et al., 2010). Besides that, the algorithm framework of ABC is relatively simple, making it possible to acquire good results at a low computational cost (Karaboga et al., 2014).

In this paper, we present an improved balance-evolution ABC (BE-ABC) algorithm to tackle the protein structure optimization problem based on the three-dimensional AB off-lattice model. The novelty behind our proposed BE-ABC algorithm is that it utilizes convergence information during the search process to strike a balance between local exploitation and global exploration as well as to manipulate the search intensity in the exploration/exploitation phases. In addition, an original overall degradation procedure is introduced as part of the algorithm to efficiently prevent premature convergence. Apart from comparing this new algorithm to some competitive algorithms from the literature (including the conventional ABC algorithm) using the well-known artificial Fibonacci sequences, we also evaluate the performance of BE-ABC on several actual protein sequences listed in the PDB database (http://www.rcsb.org/). Our numerical results indicate that BE-ABC can outperform the other approaches in the majority of the cases tested.

The remainder of this paper is organized as follows. Section 2 briefly reviews the three-dimensional AB off-lattice protein model. Then, in Section 3, we describe the conventional ABC algorithm and some issues associated with it. The details of BE-ABC are presented in Section 4, followed by Section 5 where experimental setups and results obtained through this algorithm to tackle the protein structure optimization problem are discussed. Finally, the conclusion is drawn and possibilities for future work are highlighted in the last section.

2. The three-dimensional AB off-lattice protein model

This section describes how a three-dimensional protein structure can be determined by a set of bond/torsional angle parameters and how the corresponding free-energy function value can be calculated.

According to the AB off-lattice model, the main driving forces that contribute to protein structure formulation are the hydrophobic interactions (Wang et al., 2013). It assumes that all kinds of amino acids in a protein sequence fall into two categories, namely, ones with hydrophobic residues (represented by A particles) and the others with hydrophilic residues (represented by B particles) (Kim et al., 2005). Particles are linked up sequentially by unit-length chemical bonds, and then form a chain in the three-dimensional space. Conformation of such a chain with N particles will be determined by $(2N-5)$ angle parameters $[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]_{1\times(2N-5)}$ uniquely, which consist of $(N-2)$ bond angles and $(N-3)$ torsional angles. Locations of amino acids in the three-dimensional space are determined by Eq. (1):

$$(x_i, y_i, z_i) = \begin{cases} (0,0,0) & i=1 \\ (0,1,0) & i=2 \\ (\cos(\theta_1), \sin(\theta_1)+1, 0) & i=3 \\ (x_{i-1}+\cos(\theta_{i-2})\times\cos(\beta_{i-3}), y_{i-1}+\sin(\theta_{i-2})\times\cos(\beta_{i-3}), z_{i-1}+\sin(\beta_{i-3})) & 4 \le i \le N \end{cases}. \quad (1)$$

Here, the first three amino acid particles are defined in the plane of $z=0$ for the convenience of normalization. Then, locations of the subsequent particles (i.e., when $i \ge 4$) can be calculated recursively. Specifically, the location of the ith particle will be based on the location of the $(i-1)$th one $(i \ge 4)$. Fig. 2 schematically shows how the location of a protein sequence "AABA" is determined by the

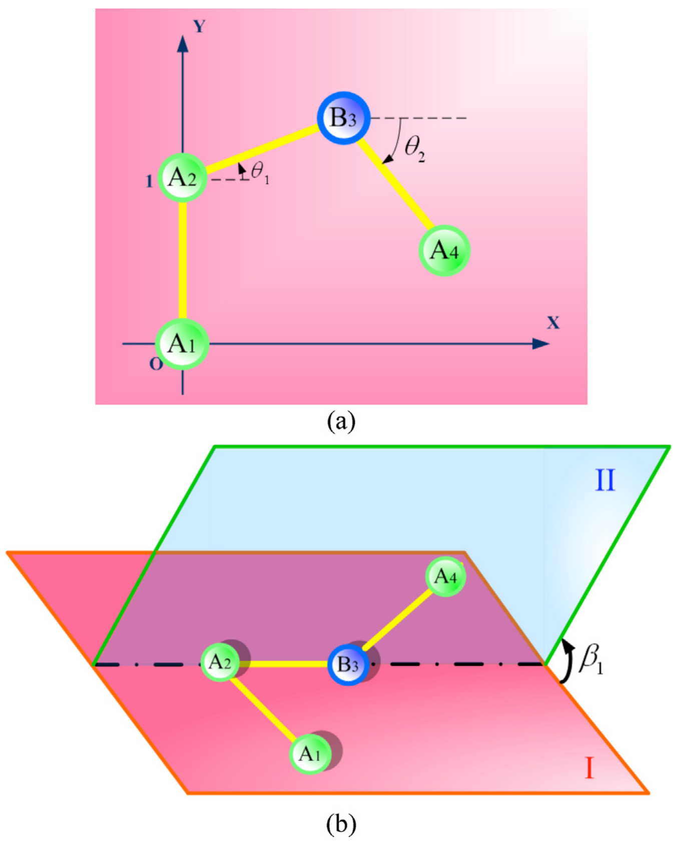

Fig. 2. A schematic diagram of the three-dimensional AB off-lattice protein model for sequence AABA (N = 4): (a) illustration of the effect of $[\theta_1, \theta_2]$ on the protein structure while $\beta_1 = 0$ and (b) illustration of the effect of β_1 on the protein structure while $[\theta_1, \theta_2]$ is temporarily fixed.

angle vector $[\theta_1,\theta_2;\ \beta_1]$, where $\angle A_1A_2B_3 = 90^\circ + \theta_1$ and $\angle A_2B_3A_4 = 180^\circ - \theta_1 - \theta_2$.

The free-energy function consists of two parts, one represents the potential energy of intermolecular interaction among amino acids while the other represents the potential energy of intermolecular interaction between the peptide chain and surrounding solvent molecules. It can be defined as follows:

$$Energy([\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]) = \sum_{i=1}^{N-2} \frac{1-\cos(\theta_i)}{4} + 4\sum_{i=1}^{N-2}\sum_{j=i+2}^{N} [r_{ij}^{-12} - C(\xi_i, \xi_j) r_{ij}^{-6}], \tag{2}$$

where ξ_i reflects the binary property of the ith particle in the sequence, r_{ij} denotes the distance between particles i and j in the three-dimensional space, and $C(\xi_i, \xi_j)$ represents the interaction between those two particles. If particle i is hydrophilic, then $\xi_i = 1$; otherwise, $\xi_i = -1$. The definitions of r_{ij} and $C(\xi_i, \xi_j)$ are given in Eqs. (3) and (4), respectively:

$$r_{ij} = \sqrt{(x_i - x_j)^2 + \left(y_i - y_j\right)^2 + (z_i - z_j)^2}. \tag{3}$$

$$C(\xi_i, \xi_j) = \frac{1}{8}(1 + \xi_i + \xi_j + 5\xi_i\xi_j). \tag{4}$$

The first sum in Eq. (2) runs over the $(N-2)$ angles $\theta_i \in [-180^\circ, 180^\circ]$ of successive bond vectors. This term is the bending energy and the coupling is ferromagnetic, i.e., it costs energy to bend the chain. The second term in the equation partially competes with the bending barrier depending on the distance between particles being nonadjacent along the chain (Bachmann et al., 2005). Through these, one angle vector $[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]$ will represent one candidate structure and $Energy$ $([\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}])$ is the corresponding free-energy value, with $\theta_i, \beta_j \in [-180^\circ, 180^\circ]$ for any $1 \le i \le N-2$, $1 \le j \le N-3$.

In this way, the original protein structure prediction scheme is transformed into a constrained numerical optimization problem. Therefore, for a given protein sequence, we can find the angle vector $[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]$ that minimizes the free-energy function $Energy(\cdot)$. In the following sections, we will first introduce the conventional ABC algorithm, and then describe the proposed BE-ABC algorithm, both of which can be used for finding the angle vector:

$$[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]^* = \arg\left(\min_{-180^\circ \le \theta_i, \beta_j < 180^\circ} (Energy([\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]))\right), \quad 1 \le i \le N-2, 1 \le j \le N-3.$$

3. The ABC algorithm

In the preceding section, a description of the three-dimensional AB off-lattice model has been presented. This section and the next focus on how to find an optimal vector $[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]^*$ with the minimal free-energy function value. In this section, we first introduce the basics of ABC. Then, we discuss some underlying issues associated with it.

3.1. Basics of ABC

The conventional ABC algorithm employs three kinds of "bees": scout bees searching for nectar sources randomly, employed bees associated with specific nectar sources, and onlooker bees that keep watch on the employed bees. Typically, half of a bee colony would consist of the employed bees and the other half the onlooker bees (Karaboga and Basturk, 2008). At an initial stage, the scout bees are set out to randomly search for nectar sources. Soon after, they become the employed bees responsible for sharing information (e.g., nectar source quality and their current locations) with other employed bees as well as the onlooker bees by means of "dancing". The onlooker bees will then randomly select the locations of the employed bees to exploit. It is worth pointing out that the locations with relatively higher quality nectar sources are more likely to be chosen by the onlooker bees for exploitation. The employed bees, on the other hand, will also randomly share their locations with others to explore possible new locations. If an employed bee finds no better nectar source than one that it has previously discovered within a certain time length, it turns into a scout bee again. Its position will be a randomly initialized location in the search space.

A location of a nectar source represents a feasible solution to the problem, and the nectar quality is reflected by the objective function value (Karaboga and Akay, 2009). Let $\mathbf{X} = (X^1, X^2, \ldots, X^D)$ represent a solution in the feasible solution space, $fun(\cdot)$ be the objective function that needs to be minimized, $rand(m,n)$ be a random number between m and n obeying the uniform distribution, and SN be the population size of a bee swarm. As aforementioned, the number of onlooker bees in a bee colony is $SN/2$, equaling that of the employed bees. In this work, a solution $\mathbf{X} = (X^1, X^2, \ldots, X^D)_{1\times D}$ refers to an angle vector $[\theta_1, \ldots, \theta_{N-2}; \beta_1, \ldots, \beta_{N-3}]_{1\times(2N-5)}$ (here $D = 2N-5$) and the objective function $fun(\cdot)$ is $Energy(\cdot)$. At first, as many as $SN/2$

scout bees are randomly initialized in the feasible solution space. Eq. (5) shows how the jth element of the ith scout bee's location $\mathbf{X}_i$ is calculated:

$$X_i^j \leftarrow X_{\min}^j + rand(0,1) \times \left(X_{\max}^j - X_{\min}^j\right), i = 1, 2, \ldots, \frac{SN}{2}, \quad j = 1, 2, \ldots, D, \tag{5}$$

where $X_{\min}^j$ and $X_{\max}^j$ denote the lower and upper boundaries of this jth element, and D denotes the dimension of a feasible solution. Thereafter, the $SN/2$ scout bees will become the employed bees and an iterated process begins from here.

In each cycle of iteration, an employed bee will share information with a randomly chosen companion and change one randomly chosen element of its location vector from X_i^j to X_i^{*j} using the following equation:

$$X_i^{*j} \leftarrow X_i^j + rand(-1,1) \times \left(X_k^j - X_i^j\right), k \in \left\{1, 2, \ldots, \frac{SN}{2}\right\}, \quad j \in \{1, 2, \ldots, D\}, k \neq \mathrm{i}. \tag{6}$$

It is necessary to note that j and k are both randomly selected integers. When all the employed bees arrive at their new nectar sources $\{\mathbf{X}_i^*, i = 1, 2, \ldots, SN/2\}$, they evaluate the quality of these new nectars and then decide whether to stay at the new location or the previous one by means of a greedy selection strategy. Specifically, if the ith employed bee finds that $fun(\mathbf{X}_i^*) < fun(\mathbf{X}_i)$, it will go to the new location $\mathbf{X}_i^*$, i.e., $\mathbf{X}_i \leftarrow \mathbf{X}_i^*$; otherwise, it remains at the previous location $\mathbf{X}_i$.

When all the employed bees have decided on their locations, a roulette selection strategy will direct the onlooker bees to select "qualified" employed bees to follow. A probability index P is calculated according to Eqs. (7) and (8) to reflect the relative quality of nectar sources at which the employed bees are located.

$$P(i) = \frac{\sum_{j=1}^{i} fitness(j)}{\sum_{j=1}^{SN/2} fitness(j)}, i = 1, 2, \ldots, \frac{SN}{2}, \tag{7}$$

$$fitness(i) = \begin{cases} \dfrac{1}{1 + fun(\mathbf{X}_i)} & \text{if} fun(\mathbf{X}_i) \geq 0 \\ 1 + |fun(\mathbf{X}_i)| & \text{if} fun(\mathbf{X}_i) < 0 \end{cases}. \tag{8}$$

Each onlooker bee will search locally around an employed bee. For some ith onlooker bee, a comparison is made between a random number $rand(0,1)$ and $P(j)$. If $P(j) \geq rand(0,1)$, this onlooker bee will search around the jth employed bee; otherwise, a comparison between $rand(0,1)$ and $P(j+1)$ will be made. Even if $P(SN/2)$ happened to be smaller than $rand(0,1)$, such a comparison process is repeated from the first employed bee's $P(1)$ until a larger $P(j)$ is found. Then, the corresponding jth employed bee will be chosen. The following equation (i.e., Eq. (9)) shows the location of the ith onlooker bee $\mathbf{Y}_i = \left(X_j^1, \ldots, X_j^{k-1}, Y_i^k, X_j^{k+1}, \ldots, X_j^D\right)$ that searches locally around the selected jth employed bee.

$$Y_i^k \leftarrow X_j^k + rand(-1,1) \times \left(X_m^k - X_j^k\right), m \in \left\{1, 2, \ldots, \frac{SN}{2}\right\}, \quad k \in \{1, 2, \ldots, D\}, m \neq j. \tag{9}$$

Note that in this equation, m and k are randomly selected integers as well. When all the $SN/2$ onlooker bees have determined their locations, a greedy selection strategy is implemented. This time, however, a comparison is made between $fun(\mathbf{X}_j)$ and $fun(\mathbf{Y}_i)$, i = 1,2, . . . ,$SN/2$. If $fun(\mathbf{Y}_i)$ is smaller than $fun(\mathbf{X}_j)$, the jth employed bee will abandon the current location $\mathbf{X}_j$ and go to $\mathbf{Y}_i$, i.e., $\mathbf{X}_j \leftarrow \mathbf{Y}_i$; otherwise, the jth employed bee remains at $\mathbf{X}_j$.

It is interesting to note that every time the greedy selection is carried out, it involves one central employed bee. In the ABC algorithm, besides the probability index P, there is another index that is associated with each of the employed bees, namely *trial*, which memorizes inefficient information that is relevant to the employed bee. Specifically, $trial(i)$ records the number of times an inefficient search is performed by the ith employed bee or any onlooker bee that searches around the ith employed bee. That is, $trial(i)$ is incremented by one each time when the condition $fun(\mathbf{X}_i^*) \geq fun(\mathbf{X}_i)$ or $fun(\mathbf{Y}_j) \geq fun(\mathbf{X}_i)$ is satisfied. At the beginning, each $trial(i)$ is set to zero. As the iteration goes on, when $trial(i)$ reaches a predefined threshold *Limit*, the ith employed bee will turn into a scout bee again with a randomly initialized location in the search space (based on Eq. (5)).

The pseudo-code of the ABC algorithm is given as follows.

Algorithm 1. The ABC Algorithm

```
1.  Set the population size $SN$, and maximum cycle number $MCN$; Set the inefficient trial
    time counter $trial(i) \leftarrow 0$ ($i$ = 1,2,..., $SN/2$);
2.  Randomly initialize locations of $SN/2$ scout bees using Eq. (5);
3.  For $iter = 1$ to $MCN$, do
4.    For $item = 1$ to $SN/2$, do % the employed bee phase
5.      Generate $\mathbf{X}^*_{item}$ for the $item$-th employed bee to search according to Eq. (6);
6.      If $fun(\mathbf{X}^*_{item}) < fun(\mathbf{X}_{item})$, then % implement the greedy selection
7.        $\mathbf{X}_{item} \leftarrow \mathbf{X}^*_{item}$, and set $trial(item) \leftarrow 0$;
8.      Else
9.        $trial(item) \leftarrow trial(item) + 1$;
10.     End if
11.   End for
12.   For $i = 1$ to $SN/2$, do % prepare for the roulette selection
13.     Calculate $P(i)$ using Eqs. (7) and (8);
14.   End for
15.   Set $item = 0$;
16.   Set $j = 1$;
17.   While $item < SN/2$, do % implement the roulette selection
18.     If $P(j) > rand(0,1)$ then % the onlooker bee phase
19.       $item \leftarrow item + 1$;
20.       Choose the $j$th employed bee to follow, and then generate $\mathbf{Y}_{item}$ using Eq. (9);
21.       If $fun(\mathbf{Y}_{item}) < fun(\mathbf{X}_j)$, then % implement the greedy selection
22.         $\mathbf{X}_j \leftarrow \mathbf{Y}_{item}$, and set $trial(j) \leftarrow 0$;
23.       Else
24.         $trial(j) \leftarrow trial(j) + 1$;
25.       End if
26.     End if
27.     $j \leftarrow j + 1$;
28.     If $j > SN/2$, then
29.       Set $j \leftarrow 1$;
30.     End if
31.   End while
32.   Collect all the indices $item$ that satisfy $trial(item) > Limit$ in an index set $\Omega$; % the
    scout bee phase
33.   If $\Omega \neq \varnothing$, then
34.     Randomly choose $k \in \Omega$;
35.     Re-initialize the location of the $k$th employed bee using Eq. (5);
36.     Set $trial(k) \leftarrow 0$;
37.   End if
38.   Memorize the best solution;
39. End for
40. Output the best solution;
```

3.2. Issues with ABC

The ABC algorithm differs from other methods of its type (e.g., nature-inspired or evolutionary algorithms (Chiong et al., 2010, 2012; Chiong, 2009)) in that it carries out a two-layer searching process, executing both the exploration and exploitation procedures in each cycle of iteration. The algorithm framework of ABC is relatively simple, making it possible to acquire good solutions at a low computational cost. However, there are also several issues or limitations associated with the algorithm.

First, the exploration procedure of the conventional ABC algorithm is deemed not “global” enough. It is notable that only the value in one dimension of a solution vector (i.e., X_i^j in $\mathbf{X}_i$ for some specific $i \in \{1, 2, \ldots, SN/2\}$) is involved in the operation described in Eq. (6). When applied to some higher dimension problems, this exploration procedure will gradually become inefficient. For example, when the dimension of a problem is 10, 10% of the elements of a candidate solution will be modified through the operation as per Eq. (6); when the dimension increases to 100, only 1% will be changed. One may naturally consider that making the number of elements in a candidate solution that are involved in the exploration procedure flexible would overcome the issue. However, we must point out that increasing the dimensions blindly does not make much sense because we cannot be sure of which among the changed dimensions contributes to the change of the objective function value.

Secondly, the search efficiency in either the exploration or exploitation phase cannot be guaranteed. In Eq. (6) or (9), a random number $rand(-1, 1)$ determines how “close” a new position is to the existing one. During the exploration phase, when we calculate a new search position, if the random number is very close to 0, such a global search is essentially equivalent to a local search. In other words, since the generation of a random number $rand(-1, 1)$ obeys the uniform distribution, the efficiency of exploration/exploitation thus becomes arbitrary, making the search process uncertain. If the amplitude of such a random number can vary according to the true search demand, the whole convergence performance can be improved.

The third issue concerns the re-initialization phase (i.e., lines 32–37 of Algorithm 1) at the end of each iteration in the conventional ABC algorithm framework. Having just one scout bee at most to be generated during the re-initialization phase limits the exploration/exploitation capability of the algorithm. In fact, it has been confirmed in some numerical experiments that directly discarding the scout bees will not necessarily deteriorate the convergence performance (Karaboga and Basturk, 2008).

Apart from the three issues raised above, we note that most of the previous studies have focused too much on trying to remedy the ABC algorithm from an “algorithmic” perspective, thus overlooking potentially useful convergence information that lies within the iteration process. If such convergence information can be effectively utilized as feedback to guide the search intensity, the convergence performance may be far more efficient. This has been proven by a number of related recent work (Li et al., 2013, 2014), in which the authors incorporated such a feedback mechanism in the conventional ABC algorithm. According to the roulette selection procedure of conventional ABC, P_i reflects the ith employed bee’s relative convergence ability. If P_i is large, the number of onlookers it would attract is expected to be large too. In other words, the onlooker bees will know which employed bees are more “qualified” through the values of P. However, apart from P, the variable *trial* (see Algorithm 1) also contains useful convergence information about the employed bees, and this has been neglected by most of the aforementioned related work.

4. The BE-ABC algorithm

BE-ABC aims to overcome the issues or limitations associated with the conventional ABC algorithm. To do so, a number of improvements have been made in both the exploration and exploitation procedures.

During the exploration phase, when the ith employed bee $\mathbf{X}_i = \left(X_i^1, \ldots, X_i^2, \ldots, X_i^D\right)$ shares information with a randomly chosen companion $\mathbf{X}_k$, the new search position is calculated by the following equation:

$$X_i^{*j} \leftarrow X_g^j + rand(-1, 1) \times \left(X_k^j - X_i^j\right) \times \mu(i), k, g \in \left\{1, 2, \ldots, \frac{SN}{2}\right\}, j \in \{1, 2, \ldots, D\}, k \neq \mathrm{i}. \tag{10}$$

where the value in the jth element X_i^j is changed to X_i^{*j}. A novel multiplier $\mu(i)$ is introduced, defined as follows:

$$\mu(i) = \frac{trial(i)}{trial(i) + trial(k)}. \tag{11}$$

The operation previously described in Eq. (6) is also modified here in Eq. (10) by taking another randomly chosen companion $\mathbf{X}_g$ into account.

As in the conventional ABC algorithm, $trial(i)$ of BE-ABC records the number of times an inefficient search is performed. However, instead of 0 the lower boundary of $trial(i)$ is set to 1. The upper boundary of *trial*, meanwhile, is set to D (i.e., $Limit = D$). Since $trial(i) \in \{1, 2, \ldots, D\}$, it is now feasible to have as many as $trial(i)$ (randomly chosen) elements in the vector $\mathbf{X}_i$ changed according to Eq. (10). That is, we randomly choose $trial(i)$ different integers from 1 to D, and then set each of them to j when performing Eq. (10).

In the exploitation phase, we add a multiplier $\mu(i)$ in a similar way as per the exploration phase. Specifically, the position of the ith onlooker bee $\mathbf{Y}_i = \left(X_j^1, \ldots, X_j^{k-1}, Y_i^k, X_j^{k+1}, \ldots, X_j^D\right)$ is calculated by the following equation when it chooses the jth employed bee $\mathbf{X}_j$ to follow:

$$Y_i^k \leftarrow X_j^k + rand(-1, 1) \times \left(X_m^k - X_j^k\right) \times \mu(j), m \in \left\{1, 2, \ldots, \frac{SN}{2}\right\}, k \in \{1, 2, \ldots, D\}, m \neq j. \tag{12}$$

where m and k are randomly selected integers, and $\mu(j)$ is defined as follows:

$$\mu(j) = \frac{trial(j)}{trial(j) + trial(m)}. \tag{13}$$

During the re-initialization phase at the end of each iteration, any $trial(i)$ that has exceeded $Limit = D$ will be reset to D (rather than 0). Before proceeding to the next iteration, the average value of *trial* (i.e., $\frac{2}{SN}\sum_{i=1}^{SN/2} trial(i)$) is compared to $\alpha_{odr} \times D$, where $\alpha_{odr} \in (0, 1)$ is a user-specified scalar. If $\alpha_{odr} \times D$ is smaller than $\frac{2}{SN}\sum_{i=1}^{SN/2} trial(i)$, the whole swarm is considered to be not working efficiently to a degree of α_{odr}. Then, as many as $round(\alpha_{odr} \times SN/2)$ (randomly selected) employed bees will be re-initialized according to Eq. (5). At the same time, their corresponding *trial* indices should be reset to 1. If $\frac{2}{SN}\sum_{i=1}^{SN/2} trial(i)$ is smaller, the current iteration is terminated directly and a new iteration will begin. In this enhanced re-initialization procedure, we do not limit it to just a single scout bee at each iteration. Instead, we accumulate the necessary convergence information and make the required change at once. We call this the overall degradation strategy.

The pseudo-code of the BE-ABC algorithm is given as follows.

Algorithm 2. The BE-ABC Algorithm

```
1.  Set the population size $SN$, overall degradation rate $\alpha_{odr}$, convergence scale parameter
    $\alpha_0$, and maximum cycle number $MCN$; Set the invalid trial time counter $trial(i) \leftarrow 1$ ($i$
    $= 1,2,\ldots, SN/2$);
2.  Randomly initialize locations of $SN/2$ scout bees using Eq. (5);
3.  For $iter = 1$ to $MCN$, do
4.    For $item = 1$ to $SN/2$, do % the employed bee phase
5.      Randomly generate as many as $trial(item)$ different integers from 1 to $D$ and
        collect them in a set $\Omega$;
6.      For $ind = 1$ to $trial(i)$, do
7.        $j \leftarrow \Omega(ind)$;
8.        Calculate $X^{*j}_{item}$ according to Eqs. (10) and (11);
9.      End for
10.     If $fun(\mathbf{X}^*_{item}) < fun(\mathbf{X}_{item})$, then % implement the greedy selection
11.       $\mathbf{X}_{item} \leftarrow \mathbf{X}^*_{item}$, and set $trial(item) \leftarrow 1$;
12.     Else
13.       $trial(item) \leftarrow trial(item) + 1$;
14.       If $trial(j) > D$, do
15.         $trial(j) \leftarrow D$;
16.       End if
17.     End if
18.   End for
19.   For $i = 1$ to $SN/2$, do % prepare for the roulette selection
20.     Calculate $P(i)$ using Eqs. (7) and (8);
21.   End for
22.   Set $item = 0$;
23.   Set $j = 1$;
24.   While $item < SN/2$, do % implement the roulette selection
25.     If $P(j) > rand(0,1)$ then
26.       $item \leftarrow item + 1$;
27.       Choose the $j$th employed bee to follow, and then generate $\mathbf{Y}_{item}$ according to
          Eqs. (12) and (13); % the onlooker bee phase
28.       If $fun(\mathbf{Y}_{item}) < fun(\mathbf{X}_j)$, then % implement the greedy selection
29.         $\mathbf{X}_j \leftarrow \mathbf{Y}_{item}$, and set $trial(j) \leftarrow 1$;
30.       Else
31.         $trial(j) \leftarrow trial(j) + 1$;
32.         If $trial(j) > D$, do
33.           $trial(j) \leftarrow D$;
34.         End if
35.       End if
36.     End if
37.     $j \leftarrow j + 1$;
38.     If $j > SN/2$, then
39.       Set $j \leftarrow 1$;
40.     End if
41.   End while
42.   If $\frac{2}{SN}\sum_{i=1}^{SN/2} trial(i) > \alpha_{odr} \cdot D$, then % implement the overall degradation strategy
43.     Randomly re-initialize as many as $round\,(!\alpha_{odr} \cdot SN/2)$ employed bees' locations
        according to Eq. (5);
44.     Set their corresponding scalars $trial(\cdot) \leftarrow 1$;
45.   End if
46.   Memorize the best solution;
47. End for
48. Output the best solution;
```

5. Experimental results and discussion

We have systematically conducted a series of simulation experiments, first on a set of artificial Fibonacci sequences (which have been widely used as benchmarks in this field) and then on several actual amino acid sequences. All the simulations were carried out in a Matlab R2013a environment and executed on a 6-core Intel Xeon CPU with 64 GB RAM running at 6×3.06 GHz under Mac OS X 10.9 Mavericks.

In the first set of experiments, we investigated the convergence performance of our proposed BE-ABC algorithm and compared it to that of the conventional ABC algorithm. In the first part of the first set of experiments, we focused on the evaluation of the initial convergence rates of the ABC and BE-ABC algorithms. We used four artificial Fibonacci sequences (see Table 1) for this evaluation. In each of the four cases, we repeated the simulations for as many as 30 times to ensure that the results are statistically relevant. As already mentioned at the end of Section 2, the two bonds of each candidate solution should be $\mathbf{X}_{\min} = (-180^\circ, -180^\circ, \ldots, -180^\circ)$ and $\mathbf{X}_{\max} = (180^\circ, 180^\circ, \ldots, 180^\circ)$. User-specific parameters were set as $SN = 40$ and $Limit = D - 1$ (the latter is only for the ABC algorithm).

Figs. 3–6 show the initial convergence performances of conventional ABC and different versions of BE-ABC (with α_{odr} equals 0.3, 0.5 and 0.9, respectively) based on the four benchmark cases. From the figures, we see that the convergence rates of the ABC algorithm gradually decrease as the number of iterations increases. BE-ABC, on the other hand, is able to further improve its performance when the number of iterations is higher. It is notable that the improved performances of BE-ABC can be observed as early as 100–200 iterations in Cases 1 and 2, while it is taking a longer amount of time to pull away in Cases 3 and 4. This is understandable as the dimension of the optimization problem D increases with the length of the to-be-folded sequence N. The overall degradation procedure of BE-ABC, which overcomes premature convergence, therefore needs more time to take effect.

Detailed numerical results of the same comparisons can be seen in Table 2, where "Best" denotes the lowest free-energy value found within the given iterations, and S.D. is the standard deviation. From the table, we observe that the best-ever results obtained by the ABC algorithm are even worse than the average values obtained by the three versions of BE-ABC in Cases 3 and 4. It is interesting to note that the best results for the four cases are always found by the BE-ABC version with $\alpha_{odr} = 0.9$. Meanwhile, the other two versions of BE-ABC have better average values. A possible reason for this may be that, when α_{odr} is set to be relatively large, the *trial* values will be large too, leading to an increased number of dimensions for the algorithm to deal with. Besides that, μ will approach 0.5 on average. Under that condition, each employed bee would not "trust" the companions sufficiently. When this is the case, they would more often search in a near-random manner. Even though this adds "diversity" to the swarm (and therefore good results can still be obtained), the overall convergence performance may be affected.

Besides the overall degradation strategy we introduced in the BE-ABC algorithm, some other improvements may also have contributed to the better results obtained. For the exploration phase, the companion g does not necessarily equal to i as in Eq. (10), which promotes the global search capability of the bee swarm. More importantly, we have increased the dimensions involved in Eq. (10) according to the index of *trial*. The rationale is that, we will gradually put less "trust" in some ith employed bee's original location $\mathbf{X}_i$ as $trial(i)$ increases. For those locations, we are more interested in finding better solutions around them than knowing which change of the dimensions makes sense.

Table 1
Details of the four Fibonacci benchmark sequences.

Cases	Artificial Fibonacci sequences[a]	Lengths
1	ABBABBABABBAB	13
2	BABABBABABBABBABABBAB	21
3	ABBABBABABBABBABABBABABBABBABABBAB	34
4	BABABBABABBABBABABBABABBABBABABBABBABABBABABBABBABABBAB	55

[a] An "A" in these sequences denotes a hydrophobic amino acid and a "B" denotes a hydrophilic amino acid.

Moreover, it is notable that $\mu(i) \in [1/(D+1), D/(D+1)] \subseteq (0, 1)$. When a $trial(i)$ value becomes larger in Eq. (11), $\mu(i)$ also gets larger, which implies that we do not expect a better location near the original location $\mathbf{X}_i$ to be found, as we have already tried $(trial(i) - 1)$ times without success around $\mathbf{X}_i$. By having such a convergence multiplier, we are able to adjust the global exploration efficiency in an adaptive manner. The scenarios for the exploitation phase will be quite similar to the exploration phase and therefore we will not discuss them here.

In the second part of the first set of experiments, we focused on the final results of the same four artificial Fibonacci cases mentioned above. Figs. 7–10 show the best structures obtained for the four cases using BE-ABC, together with their corresponding solution vectors. From the figures, we observe that the hydrophobic amino acids (i.e., the light-colored particles) tend to gather to form a single hydrophobic core, which is regarded as an intrinsic characteristic in an actual protein conformation (Kim et al., 2005; Hansmann and Wille, 2002). The structures observed in Figs. 7–9

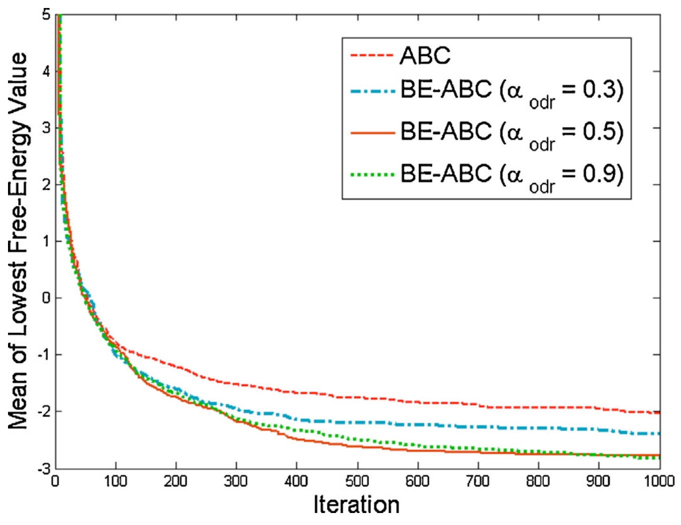

Fig. 3. Results (convergence curves) of the ABC and BE-ABC algorithms for Case 1, when $N = 13$ and $MCN = 1000$.

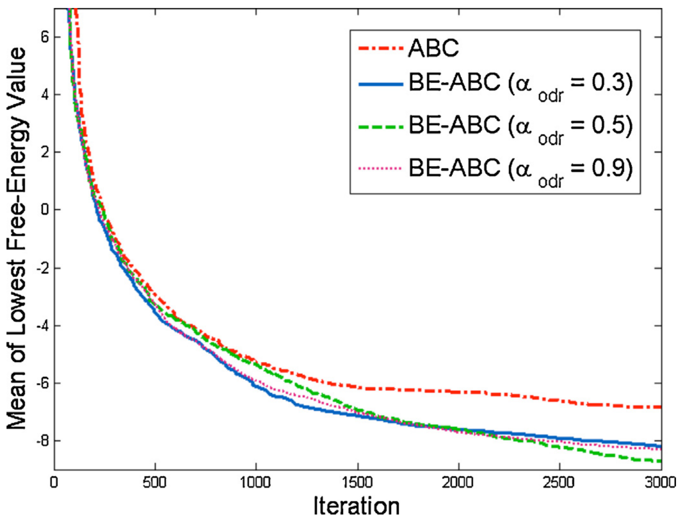

Fig. 5. Results (convergence curves) of the ABC and BE-ABC algorithms for Case 3, when $N = 34$ and $MCN = 3000$.

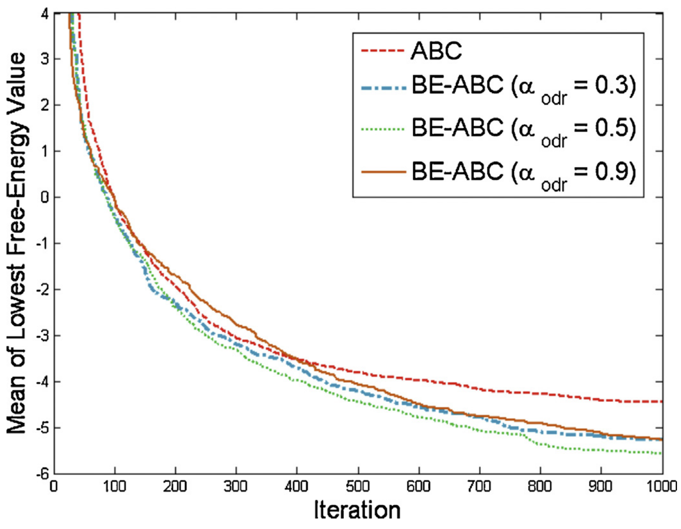

Fig. 4. Results (convergence curves) of the ABC and BE-ABC algorithms for Case 2, when $N = 21$ and $MCN = 1000$.

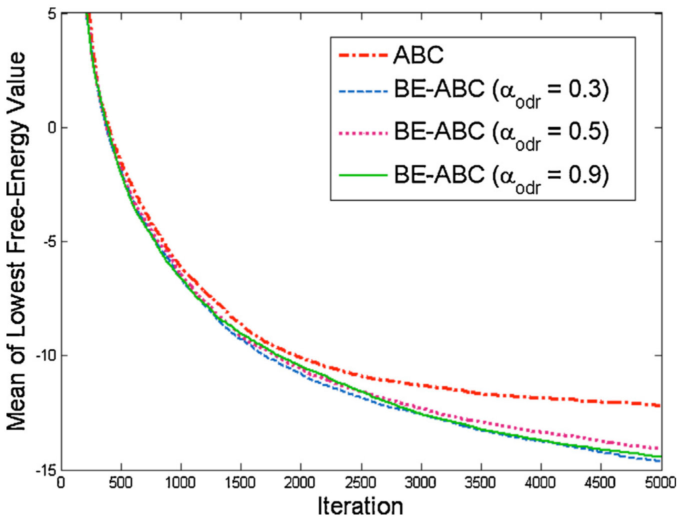

Fig. 6. Results (convergence curves) of the ABC and BE-ABC algorithms for Case 4, when $N = 55$ and $MCN = 5000$.

Table 2
Results (best, average and standard deviations) based on the free-energy values optimized by ABC and BE-ABC. The bold values denote the best results obtained for each of the four cases.

Lengths	MCN	ABC			BE-ABC ($\alpha_{odr}=0.3$)			BE-ABC ($\alpha_{odr}=0.5$)			BE-ABC ($\alpha_{odr}=0.9$)		
		Best	Average	S.D	Best	Average	S.D	Best	Average	S.D	Best	Average	S.D
13	1000	−2.9877	−2.03978	0.435489	−3.16699	−2.38128	0.328201	−3.33875	−2.78384	0.298254	**−3.3945**	−2.81963	0.382691
21	1000	−5.83668	−4.44233	0.70463	−6.18966	−5.26473	0.586174	−6.12403	−5.58228	0.466589	**−6.90646**	−5.26744	0.760577
34	3000	−7.67871	−6.8345	0.689111	−9.9171	−8.19061	1.305533	−9.9458	−8.72258	0.893539	**−10.4224**	−8.32391	0.922269
55	5000	−13.9295	−12.175	1.064592	−16.9947	−14.6346	1.214869	−17.8619	−14.061	1.38714	**−18.8385**	−14.4556	1.559396

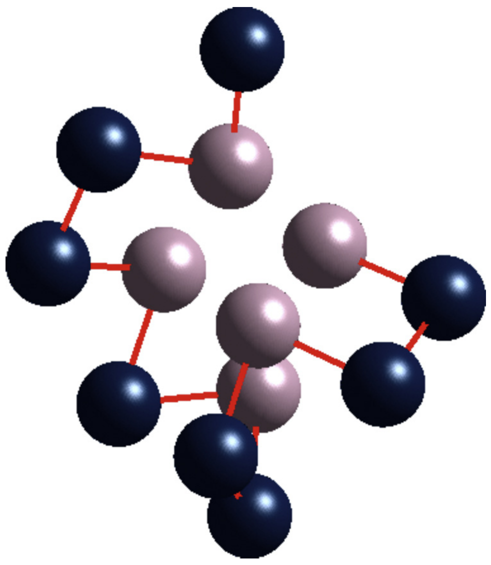
Fig. 7. The best optimized conformation of Case 1 ($N = 13$) produced by BE-ABC. The corresponding solution vector obtained is {7.6668, −83.4462, 13.1048, 0.5444, 29.1685, −47.9089, 2.7660, −31.0285, −31.2991, −46.3870, 0.2775, 9.0373, −29.5798, −116.2101, 160.5060, 0.8725, 129.3670, 24.4896, 113.3636, −161.6900, 98.6878}.

each represents a typical, if not perfect, hydrophobic core. In Fig. 10, however, the hydrophobic amino acids have formed more than one hydrophobic core. Particularly, there is a single hydrophobic amino acid located on the edge of the whole conformation, which indicates that there is still room for improvement in the optimization approach we have adopted here.

In Table 3, we show the best results obtained for the four cases by BE-ABC as well as other state-of-the-art approaches from the literature. As can be seen, apart from Case 1 the results produced by the BE-ABC algorithm are, as far as we know, currently the best. In other words, BE-ABC has achieved the best results in the literature for three of the four Fibonacci sequences.

Having confirmed the competitiveness of BE-ABC using benchmark cases, in the second set of experiments we used it to optimize the structures of three amino acid sequences taken from the real world PDB database. The detailed sequence information is given in Table 4, where I, V, L, P, C, M, A, and G are classified as hydrophobic particles “A” and amino acids D, E, F, H, K, N, Q, R, S, T, W, and Y are hydrophilic particles “B”. Table 5 shows the results (i.e., the minimal free-energy values) obtained by BE-ABC as well as other competitive algorithms from the literature that have also been used to solve the same instances. The optimized structures of these instances together with the corresponding reference structures from the PDB database are shown in Figs. 11–13.

From the figures, we can see that the directional trends and helical characteristics in the actual structures are partly

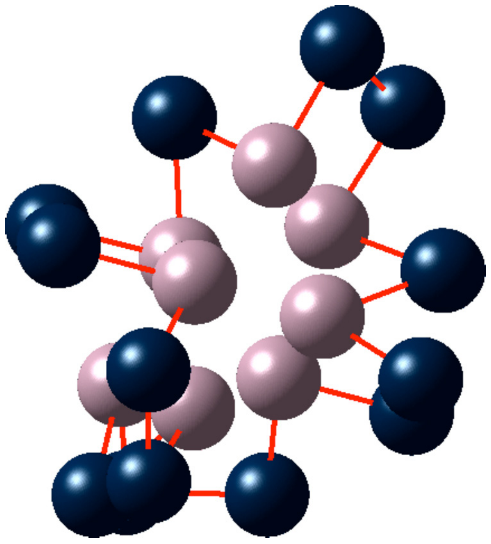
Fig. 8. The best optimized conformation of Case 2 ($N = 21$) produced by BE-ABC. The corresponding solution vector obtained is {−12.2079, −79.0870, 3.1155, −3.1035, 12.9805, 91.2166, 50.1726, −40.6936, 51.0603, −48.1108, 26.8601, 0.0720, 27.8137, −37.4555, 35.1826, −4.6902, −50.3810, −7.2643, −41.1901, 183.2030, 38.2823, 124.4475, −151.2753, −6.8236, −129.3518, 160.3034, 150.0074, -15.7557, 182.8329, −89.7776, -4.8738, −134.5553, 14.0737, −86.5181, 4.0449, 83.9872, −20.3759}.

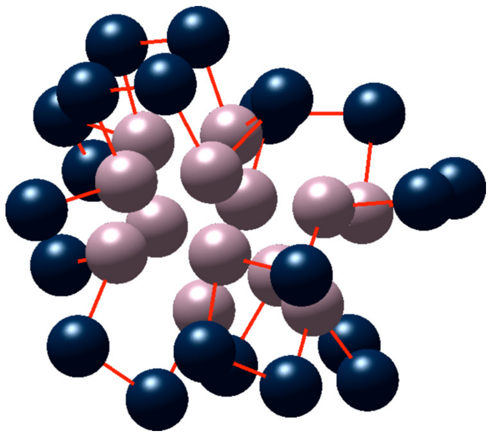
Fig. 9. The best optimized conformation of Case 3 ($N = 34$) produced by BE-ABC. The corresponding solution vector obtained is {12.5027, −83.8041, 12.2647, −103.3954, 7.8085, −89.8157, −18.7880, 94.4761, 10.2527, 90.9340, −21.9311, −64.2902, −17.6012, 67.2503, 9.0795, 75.4116, −2.9877, −99.5518, −19.9625, 37.5414, −38.6394, 8.7659, −22.6722, −55.7635, −5.7866, −53.6480, −2.1783, 59.9339, 141.2267, −7.8860, −41.9082, 41.3595, −14.9951, 27.7694, 48.3497, −163.7672, 15.6816, −156.6267, −167.6175, −34.4729, −5.0923, −50.2454, −142.5524, 141.2369, −148.9240, 102.5462, −165.9768, 141.5774, 164.2744, −129.0713, 2.1854, −167.5684, 105.9443, 22.3075, 134.2885, 34.6981, −36.5549, 60.0071, −169.9597, 44.4379, −142.2840, −50.4247, 166.9028}.

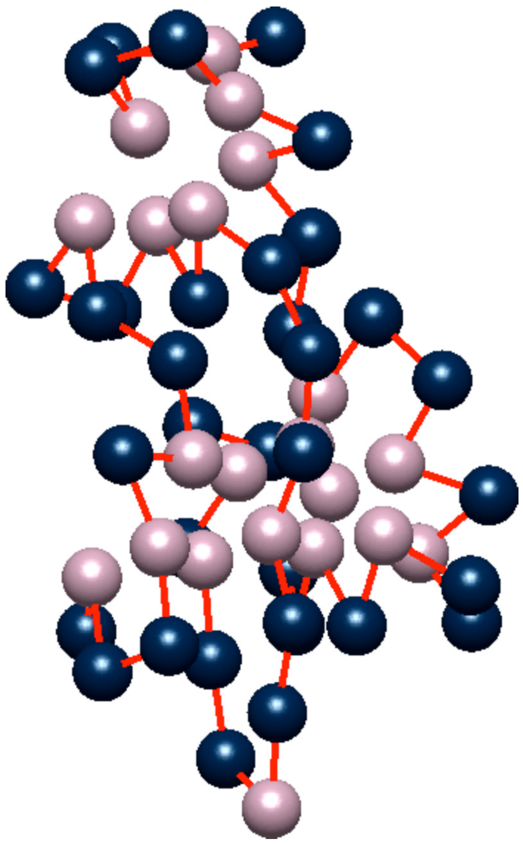

Fig. 10. The best optimized conformation of Case 4 ($N = 55$) produced by BE-ABC. The corresponding solution vector obtained is {32.9181, −29.7317, −93.7656, 14.6841, 74.8821, 51.6619, 2.2924, 34.3283, 11.9352, 146.1874, 28.0960, 3.7023, 30.0023, 43.9485, −161.3024, −10.9323, −50.8058, 31.5991, 15.2312, −16.9526, −24.1137, −46.1494, 6.1394, −12.6895, 73.3660, −22.3698, 18.2513, −39.8827, −24.5777, −8.1067, −21.4569, −2.4041, −9.5265, −31.4732, 1.6096, −50.7624, −12.8856, −12.3743, 3.6093, −51.7533, −19.3243, 28.2684, 74.8874, 0.9731, −39.1818, 11.0230, −2.2760, 84.4138, 22.8209, −36.5379, −2.7165, −38.9390, −45.4530, −143.6594, 131.0488, 158.0694, −132.9086, −153.4302, −20.5606, −129.2831, −78.4770, −26.5944, 124.1287, −148.7223, −61.3618, −159.9567, −138.5644, −146.9238, 144.6585, 28.5928, −72.7670, 39.4824, −63.6328, 19.6244, 119.3114, 13.1794, 22.0522, −165.3591, −56.1132, 173.1622, −101.1326, −102.2982, −162.3636, 94.3930, 103.8587, 47.7445, 113.9849, −10.7027, 95.6131, 50.7044, 24.3676, −74.7677, 34.8914, −62.5905, 20.6457, 129.5437, −132.0079, −156.8942, −101.8609, 12.3561, −112.4338, −114.3754, −24.9036, 56.3568, −46.5996}.

revealed in the conformations optimized by BE-ABC. Specifically, the α-helices at the bottom of the structure we optimized in Fig. 12 match the actual ones on the right of the same figure. As aforementioned, hydrophobic residues (see the light-colored particles in Figs. 11–13) tend to form a core with a minimum surface area that encounters water molecules. Therefore, such a hydrophobic core is often surrounded by hydrophilic residues. This can be easily observed in the structures we optimized using BE-ABC.

Table 4
Details of the three actual amino-acid sequences taken from the PDB database.

IDs in the PDB	Lengths	Amino acid sequences
2KPA	26	VSVDPFYEMLAARKKRISVKKKQEQP
1AGT	38	GVPINVSCTGSPQCIKPCKDQGMRFGK CMNRKCHCTPK
1AHO	64	VKDGYIVDDVNCTYFCGRNAYCNEEC TKLKGESGYCQWASPYGNACYCYKLP DHVRTKGPGRCH

Table 5
Results of the lowest free-energy values for the three actual sequences based on the AB off-lattice model achieved by tabu search (TS) (Cheng, 2009), improved particle swarm optimization (I-PSO) (Chen et al., 2011), genetic and tabu algorithm based on matrix coding (GTAMC) (Mansour, 2011), GATS (Zhang et al., 2010; Wang and Zhang, 2011), and our BE-ABC. The bold values denote the best results obtained for each of the instances.

IDs in the PDB	TS	I-PSO	GTAMC	GATS	BE-ABC
2KPA	–	−15.9988	–	−25.10033	−**25.2558**
1AGT	−44.2656	–	−46.0002	−46.0842	−**46.0852**
1AHO	–	–	–	−69.02568	−**69.0329**

Despite the similarities, we notice that there are also some variations between the optimized structures and their corresponding references. Two possible reasons would be: (1) due to the complexity of the optimization model, there are difficulties in finding the "true" global optimum; or (2) the AB off-lattice model we adopted in this work does not fully reflect the characteristics of the amino acids in a sequence. Nevertheless, the results in Table 5 indicate that BE-ABC is the best performing algorithm in the literature based on the instances considered. This points to the likelihood that the simplified AB off-lattice model might not be up to the task for real protein structure optimization. According to the AB off-lattice model, all the amino acid residues are simply classified into the hydrophilic and hydrophobic groups. This could potentially neglect the underlying differences within either of the two groups.

6. Conclusion and future work

In this paper, we have applied the BE-ABC algorithm to optimize the structures of protein in a three-dimensional space based on the AB off-lattice model. Our simulation results clearly demonstrated that BE-ABC can outperform other state-of-the-art approaches from the literature. It is therefore an effective and viable option for the protein structure optimization problem.

Despite the positive results, we hope to further enhance the performance of our approach on sequences with more amino acids (e.g., >100). Our future work will therefore examine real-world cases with longer sequences from the PDB database. Besides that, we will also investigate how the folding path of a protein sequence can be identified on the basis of the AB off-lattice model.

Table 3
Results of the lowest free-energy values for the four Fibonacci sequences based on the AB off-lattice model achieved by conformational space annealing (CSA) (Kim et al., 2005), improved tabu search (I-TS) (Zhang and Cheng, 2008), genetic algorithm-based tabu search (GATS) (Zhang et al., 2010), energy lanscape paving (ELP) (Hansmann and Wille, 2002), a heuristic algorithm (HA) proposed in Chen and Huang (2006), the pruned-enriched rosenbluth method (PERM) (Hsu et al., 2003), multicanonical (MUCA) sampling (Bachmann et al., 2005), statistical temperature molecular dynamics (STMD) simulation (Kim et al., 2007), chaotic artificial bee colony (C-ABC) (Wang et al., 2013), and our BE-ABC. The bold values denote the best results obtained for each of the cases.

Lengths	CSA	I-TS	GATS	ELP	HA	PERM	MUCA	STMD	C-ABC	BE-ABC
13	−4.9746	−6.5687	−6.9539	−4.9670	−4.9746	−4.9616	−4.9670	−4.9667	−**7.0025**	−6.99612
21	−12.3266	−13.4151	−14.7974	−12.3160	−12.0617	−11.5238	−12.2960	−12.3176	−14.9570	−**15.6258**
34	−25.5113	−27.9903	−27.9897	−25.4760	−23.0441	−21.5678	−25.3210	−25.4932	−28.0055	−**28.0516**
55	−42.3418	−41.5098	−42.4746	−42.4280	−38.1977	−32.8843	−41.5020	−42.4503	−42.2769	−**42.5814**

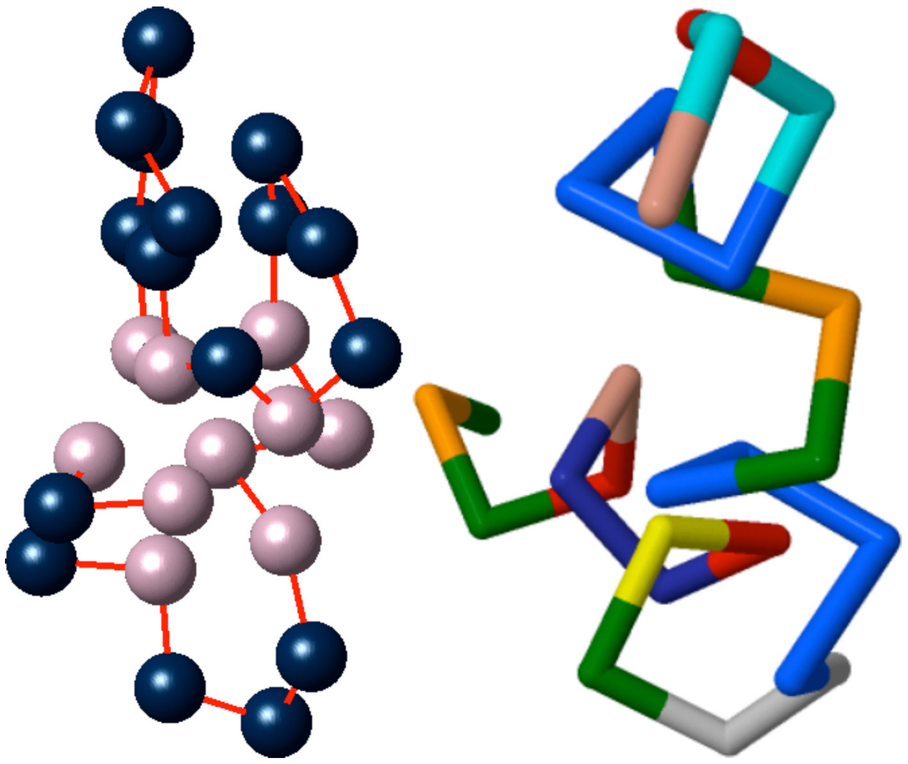

Fig. 11. The best optimized conformation of sequence 2KPA (*N* = 26) produced by BE-ABC. The corresponding solution vector obtained is {−22.7424, −80.0902, 76.3534, 4.8892, −86.1070, 24.4349, 21.9903, 30.0801, −88.8698, 59.6314, −7.9971, −15.6746, 25.0499, 38.1891, 64.2999, −59.3833, 23.9920, 18.6899, 8.7974, 25.4071, −22.0356, 17.0545, 19.6467, 4.6467, −115.0161, −177.9393, −110.7985, −24.6290, 122.7183, 64.5351, 28.3843, 6.1887, 54.9119, 78.6892, 116.9885, −147.4637, −120.5322, −47.2029, 131.3740, −25.9105, 79.7791, 138.4228, 33.6498, 19.7552, −77.7034, −78.3150, −112.8933}.

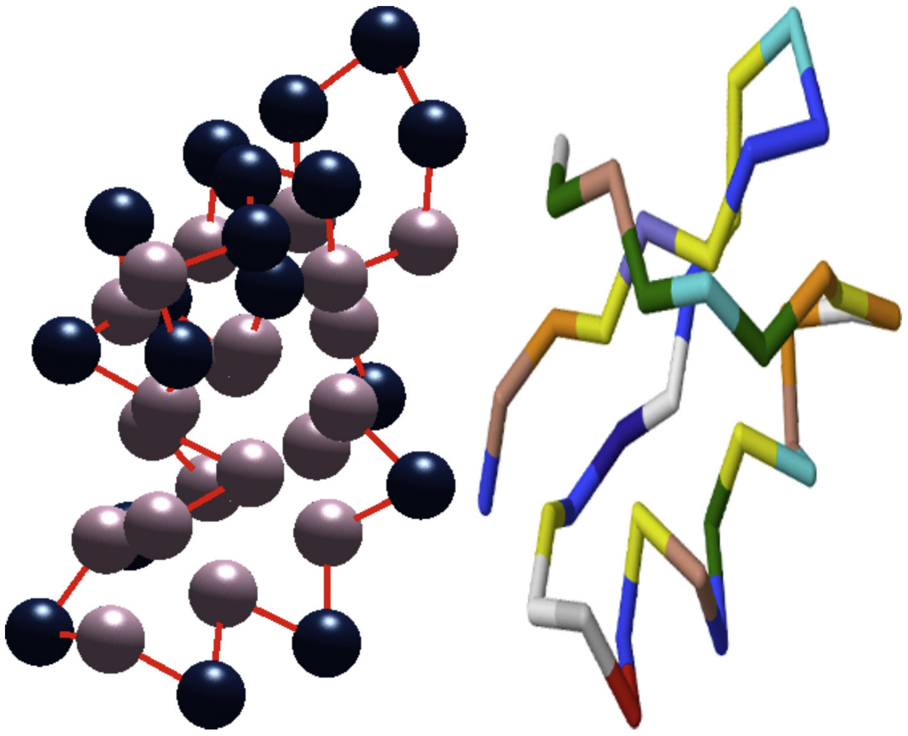

Fig. 12. The best optimized conformation of sequence 1AGT (*N* = 38) produced by BE-ABC. The corresponding solution vector obtained is {−159.1757, −123.8889, 49.0744, −54.2900, 96.1417, −168.8232, 28.1643, 136.8395, 35.0971, −37.9098, 56.3939, 6.5575, −1.6387, −67.0057, 52.1627, 172.8442, −30.4241, 80.6226, −12.2057, −45.4716, 90.0747, −22.6611, 33.8601, −54.7977, 31.5047, 6.2813, −26.5577, 25.6926, −15.4802, −32.1315, 79.7941, 31.6565, −10.4112, 33.3439, 109.0533, −22.4204, −119.8427, −134.5288, 46.8502, −142.5502, 146.0435, −15.5690, 12.3793, −19.1374, 155.3489, 36.2184, 167.5879, −113.8370, −161.8680, 103.6337, 41.4397, 124.8985, 26.1120, −12.9655, −59.2186, 168.0462, 105.4386, 178.5692, −72.8980, −117.6981, 0.1203, 56.9996, 77.5705, 122.9126, 18.5409, 178.3979, −76.4148, −104.2565, 156.6988, −22.4605, 102.0713}.

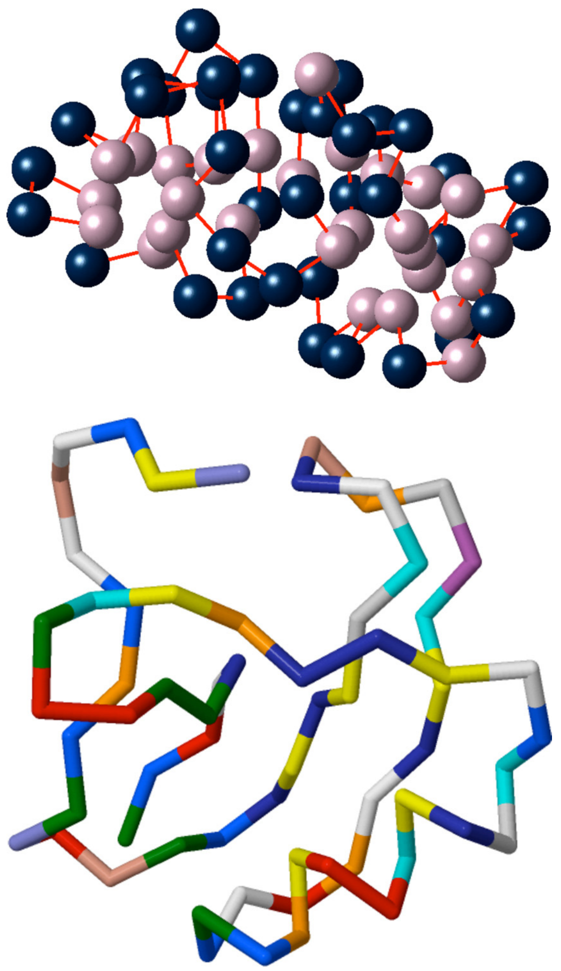

Fig. 13. The best optimized conformation of sequence 1AHO ($N = 64$) produced by BE-ABC. The corresponding solution vector obtained is {11.4938, −83.8365, −36.8448, 111.5858, 25.0645, −139.9949, −37.4185, 25.1652, 176.3678, 2.3845, 19.3226, 86.4712, −37.0268, 24.5166, −121.9576, 13.1408, −77.5472, 4.1816, −69.8191, 47.4154, −33.5796, −61.4344, 25.1123, −75.5757, 55.3962, −64.1814, −35.8398, −71.0221, 18.2894, 6.3650, −17.9972, 35.4234, 0.9697, 15.3320, −87.0670, −161.7148, −89.1391, −5.9163, −65.6728, 21.2678, −41.5493, 80.0509, 29.4580, 56.9595, 10.4166, 59.7078, −0.4628, 0.1780, −6.9465, 148.6903, −4.3582, −43.8475, −1.0512, 49.2617, −39.3322, −36.3753, −31.9670, 5.5391, 120.4185, −44.8485, −43.1804, −17.0773, −26.8270, −101.7561, 167.2774, 130.2361, −29.4028, −26.9830, 39.3165, −157.0969, 91.0763, 51.6501, 34.4031, −125.1931, −142.0274, 33.5263, 60.1433, 173.7315, −129.8505, 145.2256, 178.7261, 52.7165, 7.7540, −131.9938, −126.9691, −162.3023, −44.5222, 9.3497, 118.9008, −25.3225, 73.8177, −8.8686, −103.5647, −76.7712, 169.9265, −119.2092, −26.4241, 47.3956, −176.2411, 42.6991, −170.0001, −18.0115, 115.4736, −7.1148, −65.1082, −53.0003, 51.3744, −17.0817, 70.2470, 153.5602, 38.3858, 102.2225, 147.2370, 109.4134, −152.9103, −20.3817, −121.5138, −60.8179, −50.2070, 131.9375, −9.4940, −123.9506, −19.6806}.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the 6th National College Students' Innovation and Entrepreneurial Training Program of China under Grant No. 201210006050.

References

Anfinsen, C.B., 1973. Principles that govern the folding of protein chains. Science 181 (4096), 223–230.

Bachmann, M., Arkin, H., Janke, W., 2005. Multicanonical study of coarse-grained off-lattice models for folding heteropolymers. Phys. Rev. E 71, 31906.

Bagaria, A., Jaravine, V., Guntert, P., 2013. Estimating structure quality trends in the protein data bank by equivalent resolution. Comput. Biol. Chem. 46, 8–15.

Chen, M., Huang, W.Q., 2006. Heuristic algorithm for off-lattice protein folding problem. J. Zhejiang Univ. Sci. B 7 (1), 7–12.

Chen, X., Lv, M.W., Zhao, L.H., Zhang, X.D., 2011. An improved particle swarm optimization for protein folding prediction. Int. J. Inf. Eng. Electron. Bus. 3 (1), 1–8.

Cheng, W., 2009. Protein 3D Structure Prediction by Improved Tabu Search. Wuhan University of Science and Technology, China Ph.D Thesis.

Nature-Inspired Algorithms for Optimization. In: Chiong, R. (Ed.), Springer, Berlin.

Chiong, R., Neri, F., McKay, R.I., 2010. Nature that breeds solutions. In: Chiong, R. (Ed.), Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science and Engineering. Hershey, PA, pp. 1–24.

Variants of Evolutionary Algorithms for Real-world Applications. In: Chiong, R., Weise, T., Michalewicz, Z. (Eds.), Springer, Berlin.

Dill, K.A., 1985. Theory for the folding and stability of globular proteins. Biochemistry 24 (6), 1501–1509.

Dorn, M., e Silva, M.B., Buriol, L.S., Lamb, L.C., 2014. Three-dimensional protein structure prediction: methods and computational strategies. Comput. Biol. Chem. 53, 251–276.

Freitas, A.A., Wieser, D.C., Apweiler, R., 2010. On the importance of comprehensible classification models for protein function prediction. IEEE/ACM Trans. Comput. Biol. Bioinform. 7 (1), 172–182.

Hansmann, U.H., Wille, L.T., 2002. Global optimization by energy landscape paving. Phys. Rev. Lett. 88 (6), 68105.

Hart, W.E., Istrail, S., 1997. Robust proofs of NP-hardness for protein folding: general lattices and energy potentials. J. Comput. Biol. 4 (1), 1–22.

Hsu, H.P., Mehra, V., Grassberger, P., 2003. Structure optimization in an off-lattice protein model. Phys. Rev. E 68, 37703.

Huang, C., Yang, X., He, Z., 2010. Protein folding simulations of 2D HP model by the genetic algorithm based on optimal secondary structures. Comput. Biol. Chem. 34 (3), 137–142.

Joshi, R.R., Jyothi, S., 2003. Ab-initio prediction and reliability of protein structural genomics by PROPAINOR algorithm. Comput. Biol. Chem. 27 (3), 241–252.

Kalegari, D.H., Lopes, H.S., 2013. An improved parallel differential evolution approach for protein structure prediction using both 2D and 3D off-lattice models. Proceedings of the IEEE Symposium on Differential Evolution (SDE), Singapore, pp. 143–150.

Karaboga, D., Akay, B., 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput. 214 (1), 108–132.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39 (3), 459–471.

Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8 (1), 687–697.

Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N., 2014. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. 42 (1), 21–57.

Kim, S.Y., Lee, S.B., Lee, J., 2005. Structure optimization by conformational space annealing in an off-lattice protein model. Phys. Rev. E 72 (1), 11916.

Kim, J., Straub, J.E., Keyes, T., 2007. Structure optimization and folding mechanisms of off-lattice protein models using statistical temperature molecular dynamics simulation: statistical temperature annealing. Phys. Rev. E 76 (1), 11913.

Krishnanand, K.R., Nayak, S.K., Panigrahi, K.B., Rout, P.K., 2009. Comparative study of five bio-inspired evolutionary optimization techniques. Proceedings of the World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, pp. 1231–1236.

Li, H., Liu, K.Q., Li, X., 2010. A comparative study of artificial bee colony, bees algorithms and differential evolution on numerical benchmark problems. In: Cai, Z., Tong, H., Kang, Z., Liu, Y. (Eds.), Computational Intelligence and Intelligent Systems. Springer, Berlin Heidelberg, pp. 198–207.

Li, B., Gong, L.G., Yao, Y., 2013. On the performance of internal feedback artificial bee colony algorithm (IF-ABC) for protein secondary structure prediction. Proceedings of the Sixth International Conference on Advanced Computational Intelligence (ICACI), Hangzhou, China, pp. 33–38.

Li, B., Li, Y., Gong, L.G., 2014. Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model. Eng. Appl. Artif. Intell. 27, 70–79.

Liang, F., 2004. Annealing contour Monte Carlo algorithm for structure optimization in an off-lattice protein model. J. Chem. Phys. 120 (14), 6756–6763.

Liu, J., Xue, S., Chen, D., Geng, H., Liu, Z., 2009. Structure optimization of the two-dimensional off-lattice hydrophobic–hydrophilic model. J. Biol. Phys. 35 (3), 245–253.

Liu, J., Sun, Y., Li, G., Song, B., Huang, W., 2013. Heuristic-based tabu search algorithm for folding two-dimensional AB off-lattice model proteins. Comput. Biol. Chem. 47, 142–148.

Mansour, R.F., 2011. Applying an evolutionary algorithm for protein structure prediction. Am. J. Bioinform. Res. 1 (1), 18–23.

May, A., Pool, R., Van Dijk, E., Bijlard, J., Abeln, S., Heringa, J., Feenstra, K.A., 2014. Coarse-grained versus atomistic simulations: realistic interaction free energies for real proteins. Bioinformatics 30 (3), 326–334.

Pierce, N.A., Winfree, E., 2002. Protein design is NP-hard. Protein Eng. 15 (10), 779–782.

Poole, R.K., 2011. Globins and Other Nitric Oxide-Reactive Proteins. Academic Press, USA.

Rossi, G., Ferrando, R., 2009. Searching for low-energy structures of nanoparticles: a comparison of different methods and algorithms. J. Phys. Condens. Matter 21 (8), 84208.

Söding, J., Lupas, A.N., 2003. More than the sum of their parts: on the evolution of proteins from peptides. Bioessays 25 (9), 837–846.

Sousa, S.F., Fernandes, P.A., Ramos, M.J., 2006. Protein-ligand docking: current status and future challenges. Proteins 65 (1), 15–26.

Stillinger, F.H., 1999. Exponential multiplicity of inherent structures. Phys. Rev. E 59 (1), 48–51.

Stillinger, F.H., Weber, T.A., 1984. Packing structures and transitions in liquids and solids. Science 225 (4666), 983–989.

Stillinger, F.H., Head-Gordon, T., Hirshfeld, C.L., 1993. Toy model for protein folding. Phys. Rev. E 48 (2), 1469–1477.

Unger, R., Moult, J., 1993. Finding the lowest free-energy conformation of a protein is an NP-hard problem: proof and implications. Bull. Math. Biol. 55 (6), 1183–1198.

Venkatesan, A., Gopal, J., Candavelou, M., Gollapalli, S., Karthikeyan, K., 2013. Computational approach for protein structure prediction. Healthc. Inform. Res. 19 (2), 137–147.

Wang, T., Zhang, X.L., 2011. A case study of 3d protein structure prediction with genetic algorithm and tabu search. Wuhan Univ. J. Nat. Sci. 16 (2), 125–129.

Wang, Y., Guo, G.D., Chen, L.F., 2013. Chaotic artificial bee colony algorithm: a new approach to the problem of minimization of energy of the 3d protein structure. Mol. Biol. 47 (6), 894–900.

Zhang, X.L., Cheng, W., 2008. Protein 3D structure prediction by improved tabu search in off-lattice AB model. Proceedings of the IEEE 2nd International Conference on Bioinformatics and Biomedical Engineering (ICBBE), Shanghai, China, pp. 184–187.

Zhang, X., Wang, T., Luo, H., Yang, J.Y., Deng, Y., Tang, J., Yang, M.Q., 2010. 3D protein structure prediction with genetic tabu search algorithm. BMC Syst. Biol. 4 (Suppl. 1), 1–9.

Zhu, H.B., Pu, C.D., Lin, X., Gu, J., Zhang, S., Su, M., 2009. Protein structure prediction with EPSO in toy model. Proceedings of the IEEE Second International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, pp. 673–676.