J. of Mult.-Valued Logic & Soft Computing, Vol. 22, pp. 267–286
Reprints available directly from the publisher
Photocopying permitted by license only

Performance Analysis of Swarm Intelligence Algorithms for the 3D-AB *off-lattice* Protein Folding Problem

RAFAEL STUBS PARPINELLI[1,2], CÉSAR M.V. BENÍTEZ[2], JELSON CORDEIRO[2] AND HEITOR SILVÉRIO LOPES[2]

[1]*Applied Cognitive Computing Group, Santa Catarina State University, Brazil*
E-mail: parpinelli@joinville.udesc.br
[2]*Bioinformatics Laboratory, Federal Technological University of Paraná, Brazil*
E-mail: cesarvargasb@gmail.com, jelsoncordeiro@gmail.com, hslopes@utfpr.edu.br

Received: April 15, 2013. Accepted: July 8, 2013.

This paper compares the performance of four swarm intelligence algorithms for the optimization of a hard bioinformatic problem: the protein structure prediction problem (PSP). The PSP envolved the *protein folding* that is the process by which polypeptide chains are transformed into compact structures that perform biological functions. In this work, we tested the standard versions of the following algorithms: Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Gravitational Search Algorithm (GSA), and the Bat Algorithm (BA). The algorithms were evaluated using two criteria: quality of solutions and the processing time. The results show that the PSO algorithm presented the overall best balance between these two criteria. Also, both PSO and GSA displayed potential to evolve even better solutions, if more iterations were given.

Keywords: Swarm intelligence; 3D-AB model; protein folding problem; particle swarm optimization; artificial bee colony; gravitational search algorithm; bat algorithm

1 INTRODUCTION

Many Bioinformatics problems are featured mainly to be non-linear and strongly constrained. This is the case of the protein structure prediction problem approached in this paper. Due to the lack of exact methods for solving

such a class of problems, the need for robust heuristic methods arises. Along decades, Evolutionary Computation (EC) and Swarm Intelligence (SI) have provided a large range of flexible and robust optimization methods, capable of dealing successfully with complex optimization problems. Both EC and SI are population-based methods in which each individual of a population represents a tentative solution to the problem to be solved. In recent years several other SI algorithms have appeared inspired, for instance, by fireflies bioluminescence, slime molds life cycle, cockroaches infestation, mosquitoes host-seeking, bats echolocation, bees mating, bees foraging, and bacterial foraging [28].

In this paper, the optimization performance of four SI algorithms were tested for a hard bioinformatic problem. The empirically selected swarm-based algorithms were: Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Gravitational Search algorithm (GSA), and the Bat Algorithm (BA). All these approaches are global optimization metaheuristics and they have been used as general problem-solving methods for hard problems. However, unbiased and comprehensive comparisons between such methods for specific problems are not frequently found in the literature.

The algorithms were evaluated using two criteria: quality of solutions and the processing time.

The objective of this paper is not to propose new versions or modifications to the above-mentioned algorithms, but to verify the differences in exploitation and exploration balance for all algorithms performing an unbiased comparison over a real problem. Hence, we used the canonical versions of the algorithms set up with standard control parameters. The bioinformatic optimization problem addressed here is the protein folding problem that is strongly constrained.

2 PROTEIN STRUCTURE PREDICTION

Proteins are the basic structures of all living beings [16] and they are composed by a chain of amino acids that are linked together by means of peptide bonds. Each amino acid is characterized by a central carbon atom (referred as Cα), to which a hydrogen atom, an amine group (NH_2), a carboxyl group (COOH) and a side-chain (also known as radical R) are attached. All amino acids have the same backbone and they differ from each other by the side-chain, which can range from just a hydrogen atom (in glycine) to a complex heterocyclic group (in tryptophan). Two amino acids are linked together by the carboxyl group of one amino acid to the amino group of another [26]. Several amino acids exist in nature, but only 20 are proteinogenic and they can be

classified according to their affinity to water (hydrophobicity): hydrophobic and hydrophilic (also known as polar) amino acids.

Proteins are synthesized in the ribosome of cells following a template given by the messenger RNA (mRNA). During the synthesis, the protein folds into a unique three-dimensional structure. This process is called protein folding.

The *protein folding* is the process by which polypeptide chains are transformed into compact structures that perform biological functions. These functions include control and regulation of essential chemical processes for the living organisms. Under physiological conditions, the most stable three-dimensional structure is called the native conformation and actually allows a protein to perform its function, which, in turn, is a function of its primary structure (its linear sequence of amino acids). However, failure to fold into the intended 3-dimensional shape usually leads to proteins with different properties that simply become inactive. In the worst case, such misfolded (incorrectly folded) proteins can be harmful to the organism. For instance, several diseases such as Alzheimer's disease, cystic fibrosis and some types of cancer, are believed to result from the accumulation of misfolded proteins. That is why it is so important to study how proteins fold.

One of the most important and challenging problems in Molecular Biology with applications, such as drug design, is to obtain a better understanding of the protein folding process. In contemporary Computational Biology, there are two protein folding problems. The first problem is to predict the protein structure (conformation) from sequence (primary structure), and the second one is to predict protein folding pathways, which consists in determining the folding sequence of events which lead from the primary structure of a protein (its linear sequence of amino acids) to its native structure.

The Protein Structure Prediction (PSP) also has great practical importance in this era of genomic sequencing. Thanks to the several genome sequencing projects being conducted in the world, a large number of new proteins have been discovered. However, only a small amount of such proteins have their three-dimensional structure known. For instance, the UniProtKB/TrEMBL repository of protein sequences has currently around 23 million records (as in july/2011), and the Protein Data Bank – PDB [7] has the structure of only 83,627 proteins. This fact is due to the cost and difficulty of unveiling the structure of proteins, from the biochemical point of view.

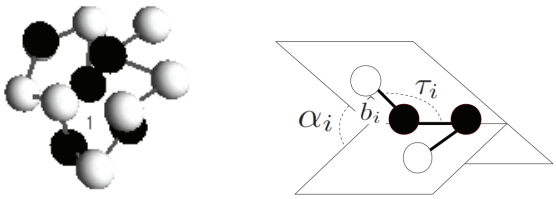
Both Physics and Computer science have an important role here, proposing models for studying the PSP problem [24]. Ideally, both the protein and the solvent should be represented at atomistic level because this approach is the closest to experiments [29]. However, the simulation of computational models that take into account all the atoms of a protein is frequently

unfeasible due to the multidimensionality of the system ($> 10^4$ degrees of freedom) [23], even with the most powerful computational resources. Consequently, several simplified models that abstract the protein structure have been proposed. The simplest computational model for the PSP problem is known as Hydrophobic-Polar (HP) model, both in two (2D-HP) and three (3D-HP) dimensions [13]. Although simple, the computational approach for searching a solution for the PSP using the HP models was proved to be NP-complete [2, 6, 10]. Other models are the three-dimensional HP Side-Chain model (3DHP-SC) [5], and the AB *off-lattice* model addressed in this work. [17, 33] and [35] employed neural networks, Monte Carlo search and biologically inspired methods using the 2D-AB off-lattice model. An extended three-dimensional version of the 2D-AB was presented by [15]. Recently, [39] introduced an improved implementation of tabu search with the 3D-AB *off-lattice* model, obtaining good performance.

2.1 The AB Off-lattice Model

The AB off-lattice model was introduced by [32] to represent protein structures. In this model each residue is represented by a single interaction site located at the Cα position. These sites are linked by rigid unit-length bonds ($\hat{b_i}$) to form the protein structure. The three-dimensional structure of an N-length protein is specified by the $N-1$ bond vectors $\hat{b_i}$, $N-2$ bond angles τ_i and $N-3$ torsional angles α_i, as shown in Figure 1. These angles are defined in the range $[-180^o, 180^o]$.

The 20 proteinogenic amino acids are classified into two classes, according to their affinity to water (hydrophobicity): 'A' (hydrophobic) and 'B' (hydrophilic or polar). This model do not describe the solvent molecules. However, solvent effects such as the formation of the hydrophobic core are taken into account through interactions between residues, according to their hydrophobicity (species-dependent global interactions).



(a) (b)

FIGURE 1
Example of a hypothetic protein structure (a) and Definition of $\hat{b_i}$, τ_i and α_i (b, Adapted from [17]). White balls represent the polar residues and black balls represent the hydrophobic residues. The backbone and the connections between elements are shown in black lines.

When a protein is folded into its native conformation, the hydrophobic amino acids tend to pack inside the protein, in such a way to get protected from the solvent by an aggregation of polar amino acids that are positioned outwards. Interactions between amino acids take place and the energy of the conformation tends to decrease. Conversely, the conformation tends to converge to its native state, in accordance with the Anfinsen's thermodynamic hypothesis [1].

The energy function of a folding is given by [17]:

$$\begin{aligned} E(\hat{b_i};\sigma) &= E_{Angles} + E_{torsion} + E_{LJ} = \\ &-k_1 \sum_{i=1}^{N-2} \hat{b_i} \cdot \hat{b_{i+1}} - k_2 \sum_{i=1}^{N-3} \hat{b_i} \cdot \hat{b_{i+2}} \\ &+ \sum_{i=1}^{N-2} \sum_{j=i+2}^{N} 4\varepsilon(\sigma_i, \sigma_j)(r_{ij}^{-12} - r_{ij}^{-6}) \end{aligned} \tag{1}$$

where
r_{ij} represents the distance between ith and jth residues; $\sigma = \sigma_0, ..., \sigma_N$ form a binary string that represents the protein sequence.

E_{Angles} and $E_{torsion}$ are the energies from bond angles and torsional forces, respectively; and are given, respectively, by Equations 2 and 3.

$$E_{Angles} = -k_1 \sum_{i=1}^{N-2} \hat{b_i} \cdot \hat{b_{i+1}} \tag{2}$$

$$E_{torsion} = -k_2 \sum_{i=1}^{N-3} \hat{b_i} \cdot \hat{b_{i+2}} \tag{3}$$

$\hat{b_i}$ represents the ith bond that joins the $(i-1)$th and the ith residues, and it is represented by the vector $\hat{b_i} = \vec{r}_i - \vec{r}_{i-1}$, and $k1 = -1$; $k2 = +1/2$.

The species-dependent global interactions are given by the Lennard-Jones potencial (E_{LJ}); for pairs of ith and jth residues separated by a distance of r_{ij}.

$$E_{LJ} = \sum_{i=1}^{N-2} \sum_{j=i+2}^{N} 4\varepsilon(\sigma_i, \sigma_j)(r_{ij}^{-12} - r_{ij}^{-6}) \tag{4}$$

Where $\varepsilon(\sigma_i, \sigma_j)$ is chosen to favor the formation of the hydrophobic core ('A' residues). Thus, $\varepsilon(\sigma_i, \sigma_j)$ is 1 for AA interactions and 1/2 for BB/AB interactions.

3 SWARM INTELLIGENCE ALGORITHMS

Swarm-based algorithms are inspired by the behavior of some social living beings, such as ants, bees, birds, and fishes. Self-organization and decentralized control are remarkable features of swarm-based systems that, such as in nature, leads to an emergent behavior. Emergent behavior is a property that emerges through local interactions among system components that is not possible to be achieved by any of the components of the system acting alone [14]. In this work we compared the performance of four swarm-intelligence algorithms. These algorithms are briefly presented in the following.

3.1 Particle Swarm Optimization

The Particle Swarm Optimization algorithm (PSO) is motivated by the coordinate movement of fish schools and bird flocks [20]. The PSO is compounded by a swarm of particles that interacts each other in a continuous search space. The position of each particle represents a potential solution to the problem being solved and and it is represented as an n-dimensional vector. In PSO, particles "fly" through the hyperdimensional search space, and changes to their positions are based on the socio-cognitive tendency of particles to emulate the success achieved by other particles. Each particle of the swarm has its own life experience and is able to evaluate the quality of own experience. As social individuals, they also have knowledge about how well their neighbors have behaved. These two kind of information corresponds to the cognitive component (individual learning) and the social component (cultural transmission), respectively. Hence, decisions of an individual are take into account both the cognitive and the social components, thus leading the population (swarm) to an emergent behavior [8, 11, 31]. The PSO is shown in Algorithm 1.

3.2 Artificial Bee Colony Algorithm

The Artificial Bee Colony Algorithm (ABC) is inspired in the foraging behavior of honey bees. ABC was first proposed by [19] for solving multi-dimensional and multi-modal optimization problems. The bees aim at discovering places of food sources (that is, regions in the search space) with high amount of nectar (good fitness values, meaning good solutions for the problem). There are three types of bees: scout bees that randomly fly in the search space without guidance, employed bees that exploit the neighborhood

Algorithm 1 Canonical PSO

```
Set parameters: n, φ_p, φ_g
for i = 1 to n do
    Initialize the positions x⃗_i and velocities v⃗_i randomly
    Evaluate fitness f(x⃗_i)
    Initialize the particle's best known position to its initial position: p⃗_i = x⃗_i
    if f(p⃗_i) is better than f(g⃗) then
        Update the swarm's best known position: g⃗ = p⃗_i
    end if
end for
while stop condition not met do
    for i = 1 to n do
        Update particles' velocity: v⃗_i = v⃗_i + φ_p * r_p * (p⃗_i − x⃗_i) + φ_g * r_g * (g⃗ − x⃗_i)
        Update particles' position: x⃗_i = x⃗_i + v⃗_i
        if f(x⃗_i) is better than f(p⃗_i) then
            p⃗_i = x⃗_i
            if f(p⃗_i) is better than f(g⃗) then
                g⃗ = p⃗_i
            end if
        end if
    end for
end while
Postprocess results and visualization
```

of their locations selecting a random solution to be perturbed, and onlooker bees that use the population fitness to select probabilistically a guiding solution to exploit its neighborhood. If the amount of nectar of a new source is higher than that of the previous one in their memory, they update the new position and forget the previous one (this is a greedy selection method). If a solution is not improved by a predetermined number of trials the food source is abandoned by the corresponding employed bee and it becomes a scout bee. The ABC algorithm attempts to balance exploration and exploitation by using the employed and onlooker bees to perform local search, and the scout bees to perform global search, respectively [36]. The ABC is shown in Algorithm 2.

3.3 Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) was created based on the law of gravity and the notion of mass interactions [9, 30]. The GSA algorithm uses the theory of Newtonian physics and its searcher agents are the collection

Algorithm 2 Canonical ABC

Set parameters: n, $limit$
Initialize the food sources $\vec{x}_i$ randomly
Evaluate fitness $f(\vec{x}_i)$ of the population
$count_i = 0$
while stop condition not met **do**
  for $i = 1$ **to** $n/2$ **do** $\{Employed\ phase\}$
    Select k, j and r at random such that $k \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., d\}$,
    $r \in [0, 1]$
    $\vec{v} = x_{ij} + r \cdot (x_{ij} - x_{kj})$
    Evaluate solutions $\vec{v}$ and $\vec{x}_i$
    if $f(\vec{v})$ is better than $f(\vec{x})$ **then**
      Greedy selection
    else
      $count_i = count_i + 1$
    end if
  end for
  for $i = n/2 + 1$ **to** n **do** $\{Onlooker\ phase\}$
    Calculate selection probability
      $P(\vec{x}_k) = \frac{f(\vec{x}_k)}{\sum_{k=i}^{n} f(\vec{x}_k)}$
    Select a bee using the selection probability
    Produce a new solution $\vec{v}$ from the selected bee
    Evaluate solutions $\vec{v}$ and $\vec{x}_i$
    if $f(\vec{v})$ is better than $f(\vec{x})$ **then**
      Greedy selection
    else
      $count_i = count_i + 1$
    end if
  end for
  for $i = 1$ **to** n **do** $\{Scout\ phase\}$
    if $count_i > limit$ **then**
      $\vec{x}_i =$ random
      $count_i = 0$
    end if
  end for
  Memorize the best solution achieved so far
end while
Postprocess results and visualization

Algorithm 3 Canonical GSA

Set parameters: $n,\ \alpha, G_0$
for $i = 1$ **to** n **do**
    Initialize the positions $\vec{x}_i$ randomly
    Initialize velocities $\vec{v}_i$ and acceleration $\vec{a}_i$ to zero
end for
while stop condition not met **do**
    Evaluate the fitness of each agent
    Update G, $best$ and $worst$ of the population
    Calculate mass (M) and acceleration ($\vec{a}_i$)
    Update velocity $\vec{v}_i$ and position $\vec{x}_i$
end while
Postprocess results and visualization

of masses. In GSA, there is an isolated system of masses. Using the gravitational force, every mass in the system can detect the situation of other masses. The gravitational force is therefore a way of transferring information between different masses. In GSA, agents are considered as objects and their performance is measured by their masses. All these objects attract each other by a gravity force, and this force causes a movement of all objects globally towards the objects with heavier masses. The heavy masses correspond to good solutions of the problem. The position of the agent corresponds to a solution of the problem, and its mass is determined using a fitness function. The ABC is shown in Algorithm 3.

3.4 Bat Algorithm

The Bat Algorithm (BA) was first presented by [37, 38] and it is inspired by the echolocation capacity of the bats. The basic idea behind the Bat Algorithm is that a population of bats (possible solutions) use echolocation for distance sensing and fly randomly through a search space updating their positions and velocities. When hunting for a prey, the rate of pulse emission can be speeded up when they fly near their prey. Also, the loudness varies from the loudest, when searching for prey, to the quietest, when homing towards the prey. Each solution is evaluated by a fitness function and the bats' flight aims at finding food/prey (best solutions). Two important parameters are the loudness decay factor (α) that works similarly as the cooling schedule in the traditional simulated annealing optimization method, and the pulse increase factor (γ) that regulates the pulse frequency. The properly update for the pulse rate (r_i) and the loudness (A_i) balances the exploitation and exploration behavior of each bat, respectively. Since the loudness usually decrease once a bat has found its prey/solution (in order to not lose the prey), the rate of pulse

Algorithm 4 Bat Algorithm (BA)

Parameters: $n,\ \alpha,\ \gamma$
Initialize the bats population $\vec{x}_i$ and $\vec{v}_i$ randomly
Define pulse frequency $\vec{f}_i$ at $\vec{x}_i$
for $i = 1$ **to** n **do**
  Initialize pulse rates r_i and loudness A_i
end for
Compute $f(\vec{x}_i)$
Find the current best $\vec{x}_*$
while stop condition not met **do**
  for $i = 1$ **to** n **do**
    Generate new solutions by adjusting:
      Frequency: $\vec{f}_i = \vec{f}_{min} + (\vec{f}_{max} - \vec{f}_{min})\beta,\ \beta \in [0, 1]$
      Velocity: $\vec{v}_i^t = \vec{v}_i^{t-1} + (\vec{x}^t - \vec{x}_*)\vec{f}_i$
      Location: $\vec{x}^t = \vec{x}_i^{t-1} + \vec{v}_i^t$
    if $rand > r_i$ **then**
      Select a solution among the best solutions
      Generate a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if $rand < A_i$ & $f(\vec{x}_i) < f(\vec{x}_*)$ **then**
      Accept the new solutions
      Increase r_i: $r_i^{t+1} = r_i^0[1 - exp(-\gamma t)]$
      Decrease A_i: $A_i^{t+1} = \alpha A_i$
    end if
  end for
  Find the current best x_*
end while
Postprocess results and visualization

emission increases in order to raise the attack accuracy. The BA is shown in Algorithm 4.

3.5 Encoding of Candidate Solutions

An important issue when using swarm intelligence approaches for a given problem is the encoding of the candidate solutions. The encoding has a strong influence not only in the size of the search space, but also in the complexity of the problem, due to the presence of unknown epistasis between variables that form the individuals (solution vectors). In this work, a given conformation of the protein is represented as a set of bond rotation and torsion angles over a three-dimensional space, as shown in Section 2.1. Considering the folding

of a protein with n amino acids, an individual will represent the set of bond rotation and torsion angles. An individual has $(2n - 5)$ variables, such that positions P_1 to P_{n-2} represent the bond rotation angles, and P_{n-1} to P_{2n-5} represent the torsion angles.

To represent the position of the amino acids, three-dimensional Cartesian coordinates are defined by a vector (x_i, y_i, z_i). The first and second amino acids of the primary structure are set at the origin and point (0, 1, 0), respectively. Next amino acids are positioned at Cartesian coordinates relative to their predecessors and obtained by 3D geometrical transformations.

4 COMPUTATIONAL EXPERIMENTS

In this section we report the experiments done for comparing the above-mentioned algorithms. All experiments were done using the same type of desktop computers with core-2 Quad processor running at 2.8 GHz, 2 GBytes of RAM, under a minimal installation of Arch Linux. All algorithms were implemented in ANSI-C programming language.

4.1 Benchmark sequences

In the experiments reported below, a total of 4 synthetic protein sequences were used. These sequences have been previously used by other researchers (for instance, [15, 18]) and they were based on the Fibonacci sequence. In Table 1, N is the number of monomers of the sequences (13, 21, 34 and 55 amino acids-long sequences).

4.2 Control Parameters

Due to the stochastic nature of the algorithms compared in this work, for each algorithm mentioned in Section 3, 20 independent runs were done with different initial random seeds. For each run, an upper-bound for the number of fitness function evaluations was established to $5,000,000$. For each algorithm, the values chosen to the control parameters are standard values commonly used in the literature.

N	Sequence
13	ABBABBABABBAB
21	BABABBABABBABBABABBAB
34	ABBABBABABBABBABABBABABBABBABABBAB
55	BABABBABABBABBABABBABABBABBABABBAB
	BABABBABABBABBABABBAB

TABLE 1
Benchmark sequences for the 3D-AB *off-lattice* model

PSO

The parameters used by the PSO algorithm are the population size ($n = 100$), the relative influence of the cognitive component ($\varphi_p = 2.05$), and the relative influence of the social component ($\varphi_g = 2.05$).

ABC

ABC uses as parameters the population size ($n = 100$), the number of food sources ($FoodNumber = 50$) and the number of iterations without improvement before it replaces a scout bee ($limit = 100$).

GSA

The parameters used in the GSA algorithm are the population size ($n = 100$), the gravitational decay ($\alpha = 20$) and the gravitational initial value ($G_0 = 100$).

BA

The parameters used in the BA algorithm are the population size ($n = 100$), the loudness decay factor ($\alpha = 0.9$), and the pulse increase factor ($\gamma = 0.9$).

5 RESULTS AND ANALYSIS

In this section the results of our experiments are presented, as well as a comparison of performance among all approaches. The performance of each approach takes into account the average best solution found over all runs and the average processing time. Results are shown in Table 2. In this table,

	ABC			**PSO**		
N	E_{avg} **(avg±stdev)**	E_{best}	$t_p(s)$	E_{avg} **(avg±stdev)**	E_{best}	$t_p(s)$
13	-24.286 ± 0.34	-24.821	94.55	-23.102 ± 0.93	-24.888	157.15
21	-40.649 ± 1.55	-43.982	225.05	-43.047 ± 2.34	-46.611	372.10
34	-58.730 ± 2.22	-64.072	550.05	-70.866 ± 5.95	-80.409	908.75
55	-80.812 ± 3.93	-89.402	1373.6	-87.715 ± 19.27	-115.758	1908.85
	GSA			**BA**		
N	E_{avg} **(avg±stdev)**	E_{best}	$t_p(s)$	E_{avg} **(avg±stdev)**	E_{best}	$t_p(s)$
13	-19.213 ± 4.87	-24.492	942.70	-21.233 ± 2.19	-24.874	279.10
21	-36.937 ± 7.01	-45.965	1758.80	-31.035 ± 4.60	-38.627	697.60
34	-44.427 ± 20.16	-75.483	3262.5	-42.771 ± 4.71	-53.364	1737.7
55	-42.735 ± 26.68	-86.856	6203.50	-47.698 ± 5.70	-53.7314	4380.80

TABLE 2
Results for the 3D AB off-lattice model.

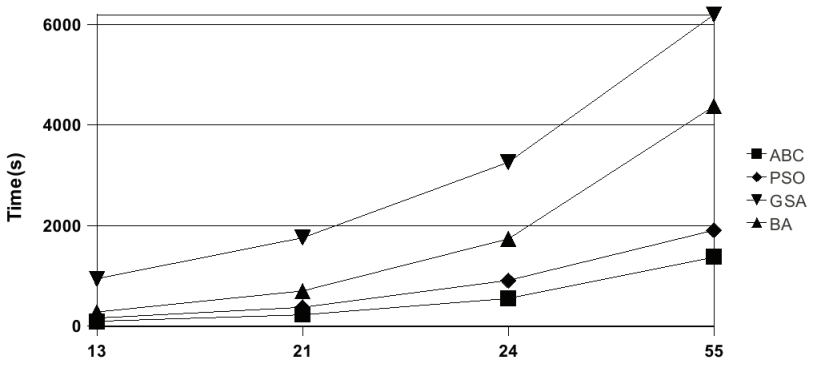


FIGURE 2
Graph showing the $t_p(s)$ of all approaches for all sequences.

columns "E_{avg}", "E_{best}" and "t_p" identify the average value of the best solutions obtained in 20 runs, the best-ever solution and the average processing time, respectively.

Concerning the computational effort, Figure 2 shows that the processing time grows exponentially as the number of amino acids of the sequence increases (particularly for the GSA and BA algorithms). This fact, by itself, strongly suggests the need for highly parallel approaches for dealing with the PSP.

Figure 3 shows the results for the average value of the best solutions obtained (E_{avg}) by all approaches for all sequences.

For the smallest sequence (13 amino-acids) all algorithms achieved almost the same performance; although the ABC performed slightly better and the

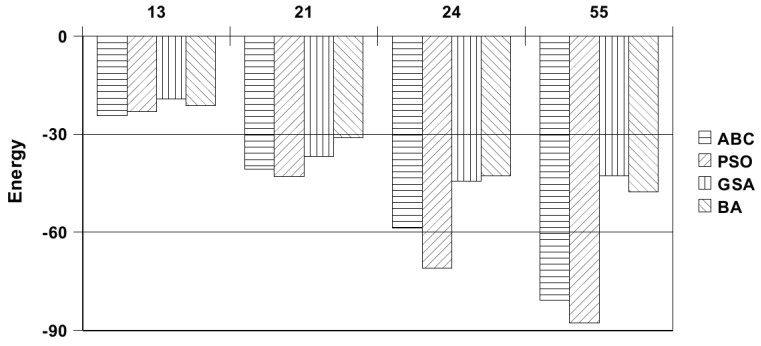


FIGURE 3
Bar graph showing the E_{avg} of all approaches for all sequences.

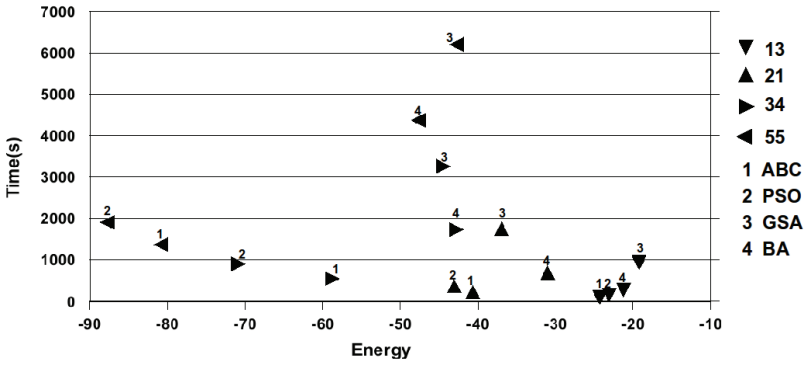


FIGURE 4
Pareto graph.

GSA performed slightly worse. For the other tree sequences (21, 24 and 55 amino-acids) the PSO and the ABC algorithms outperformed GSA and BA algorithms, with an increasing advantage towards the PSO proportionally to the sequence size. On the other hand, the GSA results outperformed the BA results for the 21 amino-acids-long sequence; both the GSA and the BA obtained almost the same results for the 24 amino-acids-long sequence; and the BA results outperformed the GSA results for the 55 amino-acids-long sequence. Overall, concerning quality of solution, the PSO algorithm obtained the best results, except for the smallest sequence.

A joint analysis of quality of solutions and computational effort was done by using the the concept of Pareto optimality [12]. A plot is constructed in such a way to represent the behavior of the two criteria to be minimized: the smaller the free energy, the better; and the smaller the processing time, the better. In other words, in this case, for each sequence the non-dominated Pareto set is located at the bottom-left corner of the Cartesian plane. In Figure 4, each point in the plot represents an amino-acids sequence, identified by different symbols and labels. For the 13 amino-acids-long sequence the non-dominated solution is achieved by the ABC algorithm. For all other amino-acids sequences, the non-dominated set includes both the PSO and ABC solutions. Analyzing all points in the plot, the non-dominated Pareto set has seven solutions (three from the PSO algorithm, and four from the ABC algorithm).

For comparisons purpose, Table 3 shows the best results currently found in the literature for the 3D AB model (“ground values”), and the percent difference between them and those obtained by the algorithms compared in this work. It is observed that our results are slightly worse than the best known results for the 13 and 21 amino-acids-long sequences. Notice that the

		diff (%)				
N	E_{best}	**[*reference*]**	**ABC**	**PSO**	**GSA**	**BA**
13	−26.507	[22]	6.568	6.299	7.901	6.354
21	−52.917	[3]	18.440	12.672	14.060	31.219
34	−97.7321	[21]	41.605	19.447	25.689	58.7272
55	−173.9803	[21]	64.225	40.188	66.803	105.614

TABLE 3
Comparison of results with the best values in the literature.

processing time needed to achieve the ground values are unknown. Taking into account that this work uses the standard versions of the algorithms, it is probable that even better results can be obtained by adjusting the control parameters of the algorithms.

Figures 5 to 8 show the convergence plot of the algorithms for all amino-acids tested. In this figure, the x-*axis* shows the number of iterations and the y-*axis* represents the best-ever value averaged over the same iteration of all runs. Analyzing these plots we can observe that for 13 and 21 amino-acids-long sequences all algorithms converged to a stagnation point. However, for 24 and 55 amino-acids-long sequences both PSO and GSA algorithms clearly could evolve to even better results if more iterations were allowed.

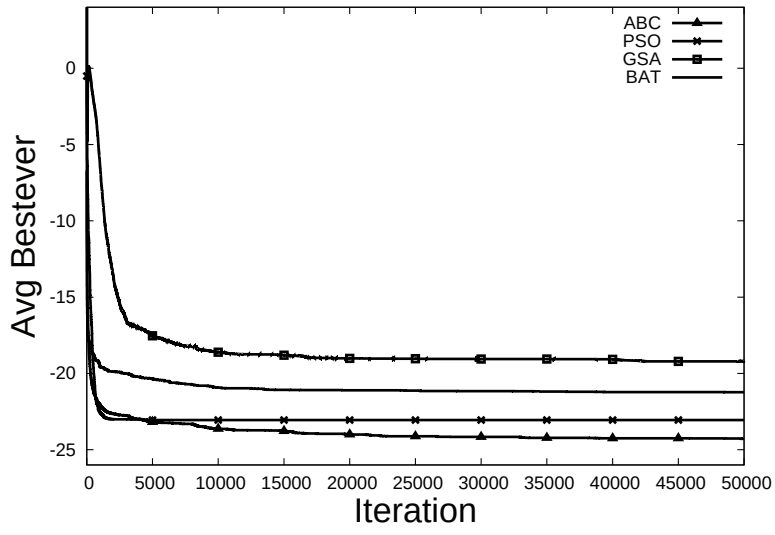


FIGURE 5
Convergence plot for the 13 amino-acids-long sequence.

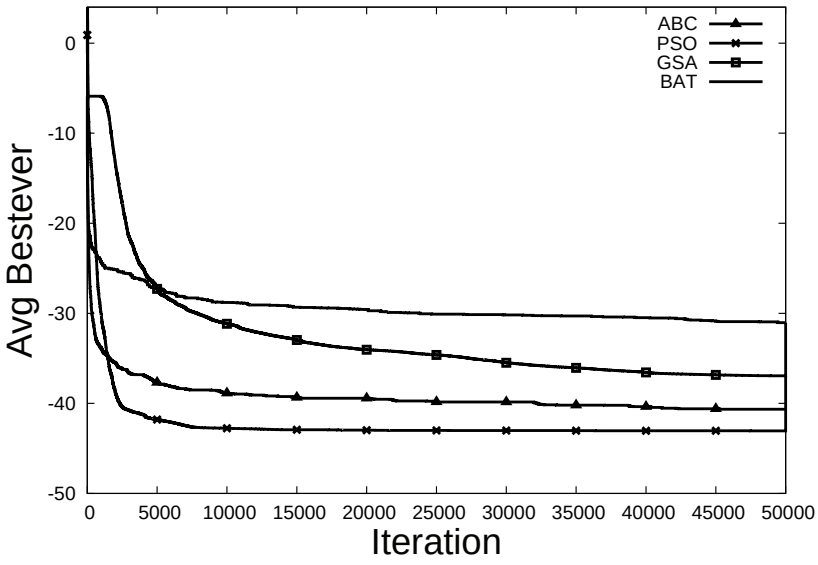


FIGURE 6
Convergence plot for the 21 amino-acids-long sequence.

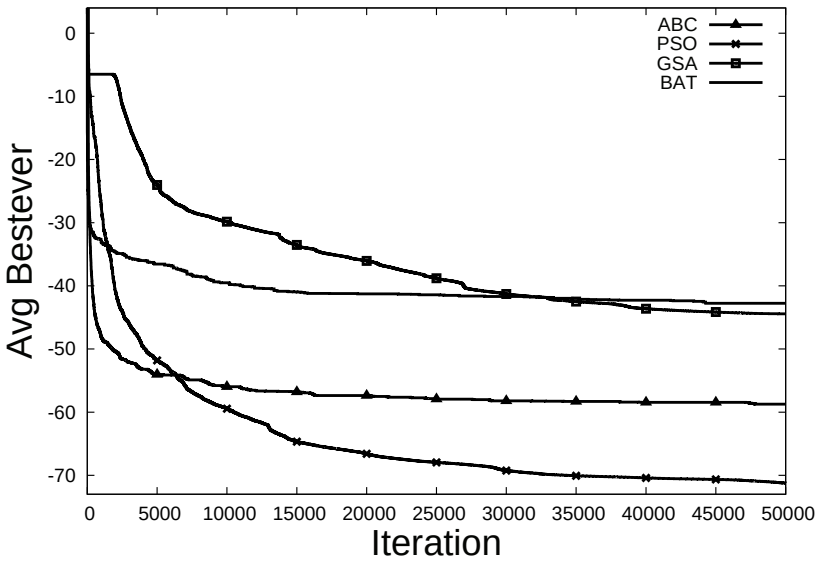


FIGURE 7
Convergence plot for the 34 amino-acids-long sequence.

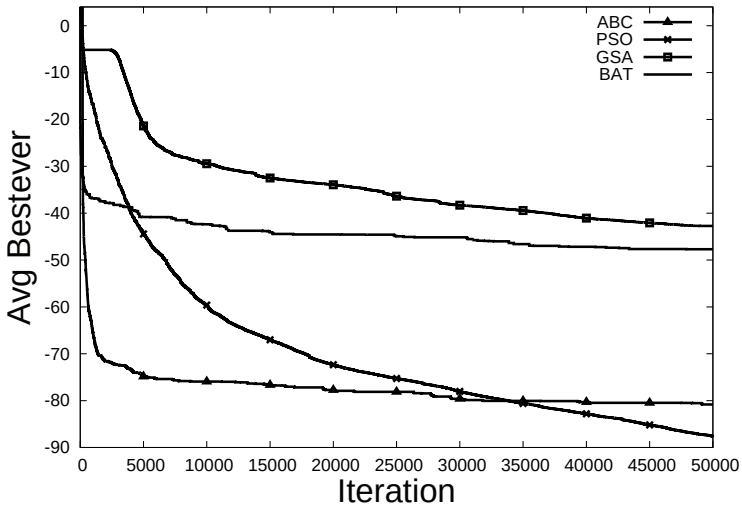


FIGURE 8
Convergence plot for the 55 amino-acids-long sequence.

6 CONCLUSIONS

The performance of four different Swarm Intelligence algorithms was analyzed in this paper, under the task of minimizing the energy function of a Protein Structure Prediction problem, featuring the off-lattice 3D-AB model. An unbiased comparison of the standard versions of the algorithms was done, taking into account not only the quality of solutions, but also, the computational effort (regarding processing time).

Significant differences were noticed, thanks to the distinctive exploitation/exploration balance of each algorithm. The obtained results pointed out the PSO algorithm as the overall best approach, especially when the complexity of the problem increases. Also, giving more iterations, both PSO and GSA algorithms could evolve to better results.

Aiming a fair comparison, we did not make any effort to tune the control parameters for the optimization algorithms. However, this is a key issue for future research in order to improve the results. Also, based on the results presented, we consider some combinations between algorithms as an alternative to solve this and other real problems more efficiently. Recent literature has indicated that the use of hybrid evolutionary systems working in a cooperative way can perform better than using single algorithms (see, for instance, [4, 25, 27] and [34]). Possible approaches for this is to form a pipeline, passing the results from one algorithm to another, or in parallel,

choosing the best result from concurrent runs of several algorithms or by having communication between the algorithms to each other favoring the co-evolution of populations.

ACKNOWLEDGMENTS

Authors would like to thank the Brazilian National Research Council (CNPq) for the research grant to H.S. Lopes; as well as to FUMDES program for the research grant to R.S. Parpinelli; and CAPES-DS scholarship to C.M.V. Benítez.

REFERENCES

[1] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. (September 1961). The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences of the United States of America*, 47:1309–1314.

[2] J. Atkins and W.E. Hart. (1999). On the intractability of protein folding with a finite alphabet. *Algorithmica*, 25(2-3):279–294.

[3] M. Bachmann, H. Arkm, and W. Janke. (2005). Multicanonical study of coarse-grained off-lattice models for folding heteropolymers. *Physical Review E*, 71:1–11.

[4] C. M. V. Benítez, R. S. Parpinelli, and H. S. Lopes. (2011). Parallelism, hybridism and coevolution in a multi-level ABC-GA approach for the protein structure prediction problem. *Concurrency and Computation: Practice and Experience*.

[5] C.M.V. Benítez and H.S. Lopes. (2010). Hierarchical parallel genetic algorithm applied to the three-dimensional HP side-chain protein folding problem. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pages 2669–2676. IEEE Computer Society.

[6] B. Berger and F.T. Leighton. (1998). Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40.

[7] H.M. Berman, J. Westbrook, Z. Feng, and G. Gilliland et al. (2000). UniProt archive. *Nucleic Acids Research*, 28(1):235–242.

[8] X. Cai, Z. Cui, J. Zeng, and Y. Tan. (2008). Dispersed particle swarm optimization. *Information Processing Letters*, 105(6):231–235.

[9] A. Chatterjee, S.P. Ghoshal, and V. Mukherjee. (2012). A maiden application of gravitational search algorithm with wavelet mutation for the solution of economic load dispatch problems. *International Journal of Bio-Inspired Computation*, 4(1):33–46.

[10] P. Crescenzi, D. Goldman, C. Papadimitrou, A. Piccolboni, and M. Yannakakis. (1998). On the complexity of protein folding. *Journal of Computational Biolology*, 5:423–446.

[11] Z. Cui and X. Cai. (2009). Integral particle swarm optimization with dispersed accelerator information. *Fundamenta Informaticae*, 95(4):427–447.

[12] K. Deb. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK.

[13] K.A. Dill, S. Bromberg, K. Yue, and K.M. Fiebig et al. (1995). Principles of protein folding - a perspective from simple exact models. *Protein Science*, 4(4):561–602.

[14] S. Garnier, J. Gautrais, and G. Theraulaz. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1):3–31.

[15] H.P. Hsu, V. Mehra, and P. Grassberger. (2003). Structure optimization in an off-lattice protein model. *Physical Review E*, 68(3):id. 037703.

[16] L. Hunter. (1993). *Artificial Intelligence and Molecular Biology*. AAAI Press, Boston, USA, 1 edition.

[17] A. Irback, C. Peterson, and F. Potthast. (1997). Identification of amino acid sequences with good folding properties in an off-lattice model. *Physical Review E*, 55(1):860–867.

[18] D. Kalegari and H.S. Lopes. (2010). A differential evolution approach for protein structure optimisation using a 2D off-lattice model. *International Journal of Bio-Inspired Computation*, 2(3/4):242–250.

[19] Dervis Karaboga and Bahriye Basturk. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8(1):687–697.

[20] J. Kennedy and R.C. Eberhart. (1995). Particle swarm optimization. In *Proc. of the IEEE Int. Conf. on Neural Networks*, pages 1942–1948, Piscataway, USA. IEEE Press.

[21] S.Y. Kim, S.B. Lee, and J. Lee. (2005). Structure optimization by conformational space annealing in an off-lattice protein model. *Physical Review E*, 72:1–6.

[22] F. Liang. (2004). Annealing contour monte carlo algorithm for structure optimization in an off-lattice protein model. *Chemical Physics*, 120:6756–6763.

[23] A. Liwo, M. Khalili, and H. A. Scheraga. (2005). Ab initio simulations of protein-folding pathways by molecular dynamics with the united-residue model of polypeptide chains. *Proceedings of the National Academy of Sciences*, 102(7):2362–2367.

[24] H.S. Lopes. (2008). Evolutionary algorithms for the protein folding problem: A review and current trends. In T.G. Smolinski, M.M. Milanova, and A-E Hassanien, editors, *Computational Intelligence in Biomedicine and Bioinformatics*, volume I, pages 297–315. Springer-Verlag, Heidelberg, Germany.

[25] A. D. Masegosa, D. A. Pelta, I. G. del Amo, and J. L. Verdegay. (2009). On the performance of homogeneous and heterogeneous cooperative search strategies. In N. Krasnogor, B. Melián-Batista, J. A. Moreno-Pérez, J. M. Moreno-Vega, and D. A. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization*, volume 236 of *Studies in Computational Intelligence*, pages 287–300. Springer.

[26] D.L. Nelson and M.M. Cox. (2008). *Lehninger Principles of Biochemistry*. W.H. Freeman, 5^{th} edition.

[27] R. S. Parpinelli and H. S. Lopes. (2011). An eco-inspired evolutionary algorithm applied to numerical optimization. In *Proceedings of the Third World Congress on Nature and Biologically Inspired Computing*, pages 473–478, Salamanca, Spain.

[28] R.S. Parpinelli and H.S. Lopes. (2011). New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1).

[29] Day R. and Daggett V. (2003). All-atom simulations of protein folding and unfolding. *Advances in Protein Chemistry*, 66:373–403.

[30] Esmat Rashedi, Hossein Nezamabadi-pour, and Saeid Saryazdi. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

[31] Tapas Si and Nanda Dulal Jana. (2012). Particle swarm optimisation with differential mutation. *International Journal of Bio-Inspired Computation*, 11(3):212–251.

[32] F.H. Stillinger and T. Head-Gordon. (1995). Collective aspects of protein folding illustrated by a toy model. *Physical Review E*, 52(3):2872–2877.

[33] F.H. Stillinger, T. Head-Gordon, and C. Hirshfeld. (1993). Toy model for protein folding. *Physical Review E*, 48(2):1469–1477.

[34] Liang Sun, Shinichi Yoshida, Xiaochun Cheng, and Yanchun Liang. (2012). A cooperative particle swarm optimizer with statistical variable interdependence learning. *Information Sciences*, 186:20–39.

[35] A. Torcini, R. Livi, and A. Politi. (2001). A dynamical approach to protein folding. *Journal of Biological Physics*, 27(2–3):181–203.

[36] R. Xiao and Z. Huang. (June 2012). An intelligent approach to the irregular polygon layout problem based on adaptive artificial bee colony algorithm. *International Journal of Computer Applications in Technology*, 43(4):295–303.

[37] Xin-She Yang. (2010). Firefly algorithm, stochastic test functions and design optimisation. *IJBIC*, 2(2):78–84.

[38] Xin-She Yang and Xingshi He. (2013). Bat algorithm: literature review and applications. *International Journal of Bio-Inspired Computation*, 5(3):141–149.

[39] X. Zhang and W. Cheng. (2008). An improved tabu search algorithm for 3D protein folding problem. *Lecture Notes in Computer Science*, 5351:1104–1109.