CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2012; **24**:635–646
Published online 12 October 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1857

SPECIAL ISSUE PAPER

Parallelism, hybridism and coevolution in a multi-level ABC-GA approach for the protein structure prediction problem

César Manuel Vargas Benitez [1], Rafael Stubs Parpinelli [1, 2] and Heitor Silvério Lopes [1,*,†]

[1]*Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR), Curitiba (PR) 80230-901, Brazil*
[2]*Applied Cognitive Computing Group, Santa Catarina State University (UDESC), Joinville (SC) 89223-100, Brazil*

SUMMARY

This paper reports the hybridization of the artificial bee colony (ABC) and a genetic algorithm (GA), in a hierarchical topology, a step ahead of a previous work. We used this parallel approach for solving the protein structure prediction problem using the three-dimensional hydrophobic-polar model with side-chains (3DHP-SC). The proposed method was run in a parallel processing environment (Beowulf cluster), and several aspects of the modeling and implementation are presented and discussed. The performance of the hybrid-hierarchical ABC-GA approach was compared with a hybrid-hierarchical ABC-only approach for four benchmark instances. Results show that the hybridization of the ABC with the GA improves the quality of solutions caused by the coevolution effect between them and their search behavior. Copyright © 2011 John Wiley & Sons, Ltd.

Received 27 June 2011; Accepted 22 August 2011

KEY WORDS: ABC algorithm; genetic algorithm; parallel processing; coevolution; protein folding

1. INTRODUCTION

Many bioinformatics problems are featured mainly to be nonlinear and strongly constrained. This is the case of the protein folding problem approached in this paper. Because of the limitations of exact methods for solving such a class of problems, the need for more robust techniques arises. Along decades, evolutionary computation (EC) and swarm intelligence have provided a large range of flexible and robust optimization methods, capable of dealing successfully with complex optimization problems. Both EC and swarm intelligence provide population-based methods where each individual of a population represents a tentative solution to the problem to be solved. This is the case, for instance, of genetic algorithms (GAs) [1], particle swarm optimization (PSO) [2], ant colony optimization (ACO) [3], artificial bee colony algorithm (ABC) [4], and many other nature-inspired methods [5].

It is known that population-based metaheuristics can explore efficiently the use of parallel and distributed computing concepts to speed up the search process. For instance, in [6], the authors analyzed different topologies of an insular parallel GA. In [7], the authors used a master-slave model to compare both synchronous and asynchronous PSO versions in an engineering problem. In [8], both master-slave and the island models of an ACO were compared using several instances of the well-known traveling salesman problem. In [9], three different techniques were applied to GAs to improve overall execution time: the first strategy is a master-slave model that exploits the natural

*Correspondence to: Heitor Silvério Lopes, Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR), Curitiba (PR) 80230-901, Brazil.
†E-mail: hslopes@utfpr.edu.br

parallelism present in the algorithm using a multiprocessor; the second strategy implements a serial GA in a dedicated reconfigurable hardware using a pipelined architecture; and the third strategy uses both pipelining and duplicated hardware modules to provide additional concurrency.

Many heuristic search methods can be found in literature using different hybrid architectures [10]. However, concerning the island model, to date there is no use of different algorithms in different islands.

In a previous work [11], the ABC was applied to the protein folding problem using two concurrent models: a master-slave and a hybrid-hierarchical. The hybrid-hierarchical concurrent model had two levels of concurrency where in the upper level, there were multiple-population coarse-grained islands, and in the lower level, there were global single-population master-slaves. The performance of the parallel models was compared with a sequential version of the ABC algorithm for some benchmark instances.

Extending the previously mentioned work, this paper presents a study of hybridization of the ABC with a GA, in a hybrid-hierarchical concurrent model. In spite of the populational feature common to these approaches, they have their own particular way to exploit and explore the search space of the problem. Hence, we investigate the application of both techniques (ABC and GA) into the hybrid-hierarchical concurrent model in order to analyze the coevolutionary influence of one method over the other. This combination aims at taking advantage, in a single approach, of the benefits of both concurrent levels (see Section 4.1) and different population-based metaheuristics.

2. POPULATION-BASED META-HEURISTICS

The main feature of a population-based algorithm is the fact that the optimization process takes place in a set of candidate solutions at each iteration. This set of solutions can be called population, swarm, school, or hive, depending on the biological inspiration employed, and each corresponding candidate solution can be an individual, a particle, a bee, or an ant.

Algorithm 1 shows a general pseudocode of a population-based algorithm. The main loop (between lines 3–7) represents the generational loop, and line 4 defines the mechanism or criterion for selecting the best solutions (i.e., survival of the fittest as in EC, or simply to discard the worst solutions). Two important characteristics of population-based algorithms, and also for metaheuristics in general, are the intensification and diversification procedures. In line 5 of the algorithm, intensification (also known as exploitation) intends to search locally and more intensively around the best solutions (i.e., a crossover procedure in GAs, or a greedy search), whereas diversification (known as exploration) leads the algorithm to explore globally the search space (i.e., a mutation procedure in GA, or a large-scale randomization).

The main drawback of these population-based techniques when facing a complex problem is the high number of function evaluations to be performed at each iteration (or generation) leading to a time-consuming process. At each iteration, all candidate solutions need to be evaluated. Because the evaluation of candidate solutions can be very complex and are repeated in the loop (see Algorithm 1); the use of parallel processing is essential for reducing the overall processing time.

1: *Initialize* the population with random candidate solutions;
2: *Evaluate* each candidate solution;
3: **while** convergence criteria is not satisfied **do**
4: Perform competitive *selection*;
5: Apply *intensification* and *diversification* procedures;
6: *Evaluate* the new pool of candidate solutions;
7: **end while**

Algorithm 1: General pseudocode of a population-based algorithm.

DOI: 10.1002/cpe

2.1. Artificial bee colony algorithm

The artificial bee colony algorithm was first proposed by [12] for solving multi-dimensional and multi-modal optimization problems. A recent work [4] compared the ABC algorithm performance against other population-based algorithms (GA, PSO, differential evolution and evolution strategies) upon several benchmark functions. Results showed that the performance of the ABC was better than or similar to the other algorithms. Another relevant work concerning the ABC algorithm analyzed the tuning of control parameters [13].

The ABC algorithm begins with n solutions (food sources) of dimension d that are modified by the artificial bees. Each solution $\vec{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is evaluated by a fitness function $f(\vec{x}_i)$, $i = 1, \ldots, n$. The bees aim at discovering places of food sources (regions in the search space) with high amount of nectar (good fitness). There are three types of bees: the scout bees that randomly fly in the search space without guidance, the employed bees that exploit the neighborhood of their locations selecting a random solution to be perturbed, and the onlooker bees that use the population fitness to select probabilistically a guiding solution to exploit its neighborhood. If the nectar amount 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 (greedy selection). If a solution is not improved by a predefined number of trials controlled by the parameter *limit*, then the food source is abandoned by the corresponding employed bee, and it becomes a scout bee. The ABC is shown in Algorithm 2. More about the ABC algorithm can be found in the repository[†].

The ABC algorithm attempts to balance exploration and exploitation using the employed and onlooker bees to perform local search, and the scout bees to perform global search, respectively.

Two remarkable features of swarm-based systems are self-organization and decentralized control that lead to an emergent behavior. Emergent behavior is a property that emerges through interactions among system components (in this case, bees), and it is not possible to be achieved by any of the components of the system by itself.

2.2. Genetic algorithms

Genetic algorithms were first introduced by [14] as a result from natural phenomena studies that could be abstracted as computational systems. GAs are probabilistic search and optimization methods inspired by the natural selection phenomenon pointed by Charles Darwin. The evolution occurs in a population of individuals (candidate solutions for a given problem) that are submitted to genetic operators such as mutation and recombination. The natural selection guides the evolution in such a way that individuals better suited to the environment (fitness landscape) have more chances to be selected for reproduction and, as a consequence, spread their genetic material to further generations.

The main features of a GA are [15]:

- To be a population-based strategy operating in a set of candidate solutions (individuals);
- To work with an encoded solution space known as genotypical space;
- It is necessary to define only the objective function (cost or reward). There is no need of any derivative information;
- They use probabilistic transitions and nondeterministic rules.

The pseudocode of a standard GA is shown in Algorithm 3, where the set of candidate solutions represent a population of $P(t)$ individuals at instant t. Each iteration of the algorithm is called a generation (represented by the index t). The stopping criteria can be defined as a maximum number of generations, after a defined period of time, or when the algorithm converges to a specific point in the search space.

[†]ABC repository: http://mf.erciyes.edu.tr/abc/

DOI: 10.1002/cpe

Parameters: n, $limit$
Initialize the food sources $\vec{x}_i$ randomly
Evaluate fitness $f(\vec{x}_i)$ of the population
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
    end if
  end for
  Memorize the best solution achieved so far
end while
Postprocess results and visualization

Algorithm 2: Artificial Bee Colony Algorithm (ABC)

$t \leftarrow 0$
Generate initial population ($P^0 = P(t)$)
Evaluate $P(t)$ $\{Calculate\ the\ fitness\ of\ each\ individual\}$
terminate $\leftarrow$ FALSE
while terminate == FALSE **do**
  $t \leftarrow t + 1$
  Select $P(t)$ from $P(t-1)$ taking the *fitness* of each individual
  Recombinate and Mutate $P(t)$ $\{Apply\ genetic\ operators\}$
  Evaluate $P(t)$ $\{Calculate\ the\ fitness\ of\ each\ individual\}$
  Replace the old population by the new population
  if stopping criteria not reached **then**
    terminate $\leftarrow$ TRUE
  end if
end while

Algorithm 3: Pseudo-code for a standard GA

DOI: 10.1002/cpe

3. PROTEIN STRUCTURE PREDICTION PROBLEM

Proteins are complex molecules that are essential to all life forms. They are composed by chains of amino acids (there are 20 different proteinogenic amino acids), joined together by peptidic bonds. They are responsible for many biological functions in any organism. Such functions are determined by the way they are folded into specific three-dimensional structures, known as native conformations. The three-dimensional shape, in turn, is a function of its primary structure (linear sequence of amino acids) [16]. Failure to fold into the intended three-dimensional shape usually leads to proteins with different properties that can become inactive or even harmful to the organism. Therefore, better understanding the protein folding process can result in important medical advancements and development of new drugs. Only a small amount of such proteins have its three-dimensional structure known. This fact is caused by the cost and difficulty in unveiling the structure of proteins, from the biochemical point of view. This is an important motivation for developing computational methods for predicting the structure of these proteins. Computer science has an important role here, proposing models for studying the protein structure prediction (PSP) problem.

3.1. The 3D-HP side-chain model

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 [17]. Although simple, the computational approach for searching a solution for the PSP using HP models was proved to be NP-complete [18]. This fact has motivated the development of many EC approaches for dealing with the problem [19].

The HP model divides the 20 standard amino acids into only two classes, according to their affinity to water: hydrophilic (or polar) and hydrophobic. When a protein is folded in its native conformation, the hydrophobic amino acids tend to group together in the inner part of the protein, in such a way to get protected from the solvent by the polar amino acids that are preferably positioned outwards. Therefore, a hydrophobic core is usually formed, especially in globular proteins. In this model, the folding or conformation of a protein is represented in a lattice, usually square (for the 2D-HP) or cubic (for the 3D-HP). Both 2D-HP and 3D-HP models have been frequently explored in the recent literature [19].

Because the expressiveness of the HP models is very poor, from the biological point of view, a further improvement is to include a bead to represent the side-chain (SC) of the amino acids [20]. Therefore, a protein is modeled by a backbone, common to any amino acid, and an SC, either hydrophobic (H) or polar (P). The SC is responsible for the main chemical and physical properties of specific amino acids.

To compute the energy of a conformation, Li *et al*. [20] proposed an equation that considers only three types of interactions (not making difference between types of SCs). In this work, we use a more realistic way to compute the energy of a folding that accounts for all possible types of interactions [21], as shown in Equation 1.

$$\begin{aligned} H = {} & \epsilon_{HH} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{HH}} + \epsilon_{BB} \cdot \sum_{i=1,j>i+1}^{n} \delta_{r_{ij}^{BB}} + \epsilon_{BH} \cdot \sum_{i=1,j\neq i}^{n} \delta_{r_{ij}^{BH}} + \epsilon_{BP} \cdot \sum_{i=1,j\neq i}^{n} \delta_{r_{ij}^{BP}} \\ & + \epsilon_{HP} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{HP}} + \epsilon_{PP} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{PP}} \end{aligned} \tag{1}$$

In this equation, ϵ_{HH}, ϵ_{BB}, ϵ_{BH}, ϵ_{BP}, ϵ_{HP}, and ϵ_{PP} are the weights of the energy for each type of interaction, respectively: hydrophobic side-chains (HH), backbone-backbone (BB), backbone-hydrophobic side-chain (BH), backbone-polar side-chain (PH), hydrophobic-polar side-chains (HP), and polar side-chains (PP). In a chain of n amino acids, the distance (in the three-dimensional space) between the i^{th} and j^{th} amino acid interacting with each other is represented by r_{ij}^{**}. For the sake of simplification, in this work, we used unity distance between amino acids ($r_{ij}^{**} = 1$). Therefore, δ is an operator that returns 1 when the distance between the i^{th} and j^{th} elements (either backbone

DOI: 10.1002/cpe

or SC) for each type of interaction is the unity, or 0, otherwise. We also used an optimized set of weights for each type of interaction, defined by [21].

During the folding process, interactions between amino acids take place, and the free 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 [22]. In this work, we consider the symmetric of H such that to transform the problem to a maximization.

It is known that the number of hydrophobic contacts (HnC) is inversely proportional to the free-energy of a given conformation. Therefore, any algorithmic procedure for the PSP that maximizes the HnC will, conversely, take the molecule to the smallest possible free-energy state [17].

4. METHODOLOGY

4.1. Parallel hybrid hierarchical model

The hybrid hierarchical model proposed previously in [11] and used in this work has two levels: at the upper level, it has multiple-population coarse-grained islands, and at the lower level, it has single-population master-slaves. In this model, there are more than two populations working independently with a migration policy between them. For each population, there is a master process that divides the computational effort to various slave processes. Figure 1 illustrates the hybrid hierarchical model with four populations at the upper level (ellipses with solid lines) and three slaves in each population at the lower level (ellipses with dashed lines). The hybridization is achieved by allocating two islands to GA and two islands to ABC, and letting solutions migrate between them (Section 5 explains better this setup).

4.2. Solution encoding and fitness function

The encoding of the candidate solutions (bees in the ABC, and chromosomes in the GA) have to be carefully implemented, because it can have a strong influence not only in the size of the search space but also in the hardness of the problem. In the model we used for the PSP, a solution represents the spatial position of the amino acids chain in a cubic lattice (3D), using internal coordinates [19]. In this coordinates system, a given conformation is represented by a set of movements of a given amino acid relative to its predecessor in the chain. Therefore, for a protein with n amino acids, a vector of $n-1$ elements will be necessary to represent a folding. Such representation of a possible solution in an artificial bee/chromosome is called here a genotype (the same terminology was used for both ABC and GA).

As mentioned before, in the 3DHP-side chain model, the amino acids of the protein are represented by a (BB) and an SC, either (H) or (P). In the three-dimensional space, there are five

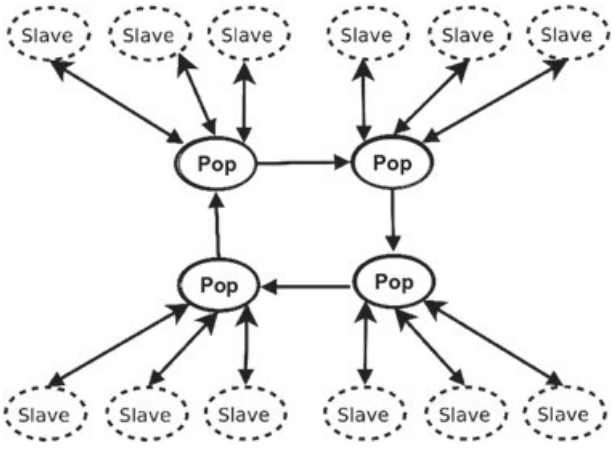

Figure 1. Hierarchical model.

DOI: 10.1002/cpe

possible relative movements for the backbone (**L**eft, **F**ront, **R**ight, **D**own, **U**p), and other five for the SC, relative to the backbone (**l**eft, **f**ront, **r**ight, **d**own, **u**p). The combination of possible movements for backbone and SC gives a set of 25 possibilities. Instead of the traditional binary alphabet, a set of 25 numbers and letters was used to encode the chromosome, as suggested by [23].

To represent the position of the amino acids in the cubic lattice, the Cartesian coordinates of each element (backbone and SC) are defined by a vector (x_i, y_i, z_i). This vector is obtained from the relative movement of an amino acid and position of its predecessor. A folding begins in the origin of the coordinates system, such that the first backbone is at $(0, 0, 0)$ and its SC at $(0, -1, 0)$. The positions of the remaining amino acids are computed following the movements encoded in the genotype of the artificial bee. Therefore, a progressive sequential procedure is necessary for decoding the genotype into a phenotype (the spatial conformation of the protein). Figure 2 shows an example of genotype–phenotype decoding, where only the first four movements are shown, because of clarity.

The fitness function used in this work was first proposed by [24] and later adapted to the 3DHP-SC model [23]. This function has three terms (as shown in Equation 2). The first one is relative to the free energy of the conformation H (computed by Equation 1), the number of collisions NC (number of points in the three-dimensional lattice that is occupied by more than one element, either BB or SC) and a penalty weight (*PenaltyValue*). The following terms represent the gyration radius of the hydrophobic and hydrophilic SCs, respectively. Radius of gyration is a measure of compactness of a set of points (in this case, the SCs of the amino acids in the lattice) [25]. This is done in such a way to favor conformations in which hydrophobic SCs are compacted within the core, and polar SCs are pushed outwards of the conformation. Equations 3, 4, and 5 show how these terms are computed.

$$fitness = [H - (NC \cdot PenaltyValue)] \cdot RadiusG_H \cdot RadiusG_P \tag{2}$$

$$RadiusG_H = maxRG_H - RG_H \tag{3}$$

$$RadiusG_P = \begin{cases} 1 & \text{if } (\mathrm{RG_P} - \mathrm{RG_H} \geq 0) \\ \frac{1}{1-(RG_P - RG_H)} & \text{else} \end{cases} \tag{4}$$

$$RG_{aa} = \sqrt{\frac{\sum_{i=1}^{N_{aa}} [(x_i - \overline{X})^2 + (y_i - \overline{Y})^2 + (z_i - \overline{Z})^2]}{N_{aa}}} \tag{5}$$

where RG_{aa} can be either RG_H or RG_P, when considering only hydrophobic or polar amino acids; x_i, y_i and z_i are the coordinates of the i-th SC of type "*aa*" of the protein, either H of P; $\overline{X}$, $\overline{Y}$ and $\overline{Z}$ are the average of all x_i, y_i, and z_i; and N_{aa} is the number of SCs of type "*aa*".

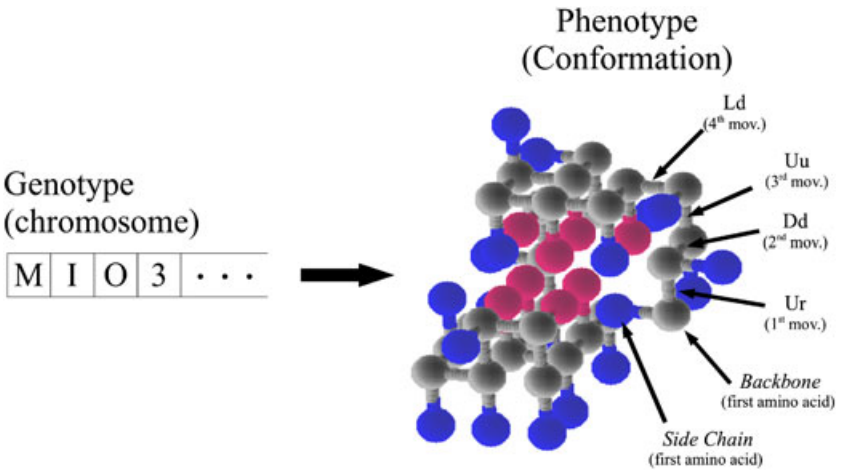

Figure 2. Example of genotype–phenotype decoding.

DOI: 10.1002/cpe

4.3. Initial population of ABC and GA

Despite the advantages of using the proposed genotypical representation, it allows two or more elements (BB or SC) to occupy the same position in the lattice. This fact is known as collision, and results in an invalid conformation, because it is physically unfeasible. However, when the initial population is randomly created, the NC tend to increase as the size of the protein increases [19]. Consequently, in the generation of the initial population, there is no guarantee that a valid individual will be generated. Therefore, not only the ABC but also the GA will spend a reasonably large time throughout the first generations working with invalid individuals until good individuals appear. Aiming at improving the performance of both algorithms, a method for creating better initial individuals was implemented based on the previous work presented in [23]. The initial population is divided into two parts (80% and 20%). The largest part representing 80% of the population is randomly generated, as usual, and the smaller part representing 20% of the population is composed only of collision-free individuals, generated by a backtracking strategy, explained next.

A folding is represented as a directed graph structured as a tree. Conceptually, each node of the tree represents a partial candidate solution c, from the first amino acid of the chain up to the last amino acid being considered. Therefore, leaf nodes represent a complete folding. Each edge of the graph represents the movement of an element (BB of SC) relative to its predecessor. The first symbol in the solution vector is decoded yielding a position in the lattice.

After positioning the first backbone and its SC in the lattice, the movement of the next amino acid backbone is randomly selected. If the movement leads to a collision with the backbone or the SC of any other amino acid previously positioned in the lattice, a backtracking is done. Other possible positions for the BB and SC are examined until a suitable combination is found (with no collisions). If this is not possible, the last pair BB/SC is removed from the current position of the lattice and set to another position. The procedure is recursively repeated until a complete valid folding is obtained. This procedure tends to be computationally intensive as the length of the protein increases. This is the reason for using such procedure with parsimony (in only 20% of the initial population).

A recurrent issue in EC algorithms that cannot be neglected here, is the use of adequate random number generators [26]. The random number generator used in our implementation was the Mersenne twister [27], which is known as one of the best generators for this purpose.

4.4. Improvement strategy

After several generations, the ABC and GA algorithms converge and can get trapped to a suboptimal region in the search space. In this case, a decrement of population diversity usually takes place. This is observed by the stagnation of the best solution for a certain number of continuous generations. When the population of solutions is concentrated around a local maxima, the only way to avoid useless computational effort is to escape from the current region and explore other regions of the search space.

In this work, we used the Decimation-and-Hot-Boot (DHB) strategy [28–30], explained next. The strategy used verifies whether or not the best-so-far solution is improved from a given generation to the next one. If it is improved, a counter is zeroed; otherwise, it is incremented. When the counter reaches a predefined number of generations, 50% of the population is decimated and substituted by individuals generated according to the same procedure done for the initial population (see section 4.3). It is important to note that the best solution is always maintained during the DHB procedure.

The application of this strategy improves the population diversity and giving chance for the evolutionary process to continue. Hopefully, this strategy can contribute to find even better solutions.

It should be taken into account that new individuals of the GA recently created by the DHB procedure probably will have low fitness values compared with those individuals that survived decimation. Therefore, it is necessary to decrease the selective pressure during some generations just after the decimation, in such a way that individuals can have the opportunity to evolve. This is accomplished by decreasing the tournament size to 2 during a fixed number of generations, and then returning to its original value.

DOI: 10.1002/cpe

5. PARALLEL IMPLEMENTATION

Two parallel approaches for the ABC were presented in a previous work [11]: a master-slave and a hybrid-hierarchical (HH–ABC). The performance of the parallel models was compared with a sequential version for four benchmark instances. Results showed that the parallel models achieved a good level of efficiency and, thanks to the coevolution effect, the HH–ABC approach could improve further the quality of solutions found.

A step ahead of development presented here, is a hybrid-hierarchical ABC-GA (HH–ABC-GA). As before, Figure 1, this approach has two levels. In the upper level, there are multiple-population coarse-grained islands. In the lower level, there are global single-population master-slaves. This combination aims at taking advantage of the benefits of both models and population-based metaheuristics in a single approach. At the lower level, there is a master process that distributes the computational effort into several slaves. In the upper level, each population (master and corresponding slaves) is seen as an island that works independently. A migration policy defines the sporadic migrations that occur between islands. It has four parameters: migration gap (number of generations between successive migrations), migration rate (number of bees that will migrate at each migration event), selection/substitution criteria for migrants, and topology of connectivity between islands. The migration policy parameters were set as: migration gap = 120 generations, migration rate = five individuals, best and four random immigrants replace random individuals of the receiving population. The topology has four islands connected by a unidirectional ring, as shown in Figure 1, but with two ABC islands (ABC_1 and ABC_2) interleaved by two GA islands (GA_1 and GA_2).

In this work, the following parameters of the ABC algorithm were adjusted using a benchmark sequence: colony size (*colonysize*=500), maximum cycle number (MCN=3000), and the percentage of employed (n_e=50%) and onlooker bees (n_o=50%). The basic parameters of the GA are: population size (*popsize*=500), tournament size (*tourneysize*=3%), crossover rate (*pcross*= 80%), and mutation rate (*pmut*=8%).

6. COMPUTATIONAL EXPERIMENTS AND RESULTS

The experiments reported in this work were run in a cluster Beowulf [31] of networked computers with 124 processing cores, running Linux. The software was developed in ANSI-C programming language, using the Message Passing Interface (MPI) MPICH2 package for the communication between processes [32] [‡].

6.1. Migration and coevolution

At the upper level, the HH–ABC-GA model uses a topology with four islands connected by a unidirectional ring. This number of islands was set because of two facts: availability of hardware (allocating one processing core per process) and the expectance of observing the coevolution effect as soon as possible. From the literature of parallel evolutionary algorithms, it is known that the topology is an important factor in the performance of the algorithm because it determines how fast a good solution disseminates to other islands. A small number of islands in a ring topology accelerate the coevolution process, thus requiring a reduced number of cycles/generations.

The effect of coevolution is illustrated in Figure 3. Immigrants inject diversity in the population where they have just arrived, thus allowing a more effective search. This figure shows the fitness value of the best individual at each population, between generations 0 to 800. The migration procedure that occurs at each 120 generations (migration gap) can be viewed at generations 120, 240, 360, 480, 600, and 720. When an individual of good quality arrives to a population, it not only improves the local best solution but also, through recombination, induces a further improvement of the quality of the population.

The HH–ABC-GA was executed and compared with the HH–ABC with same migration policy, Figure 4 shows a plot of the average fitness of the best individual (Bestever) for the two situations, at

[‡]Available at: http://www.mcs.anl.gov/research/projects/mpich2/

DOI: 10.1002/cpe

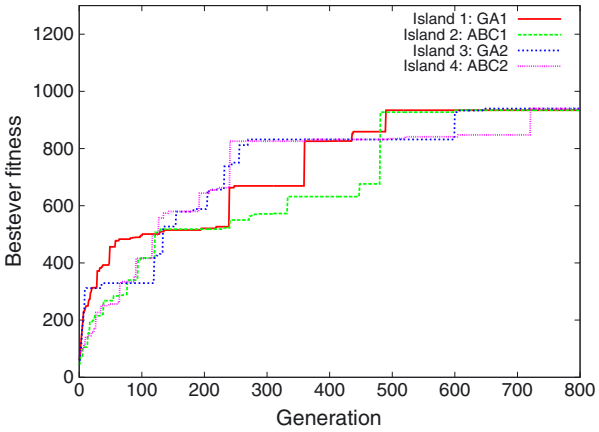

Figure 3. Coevolution.

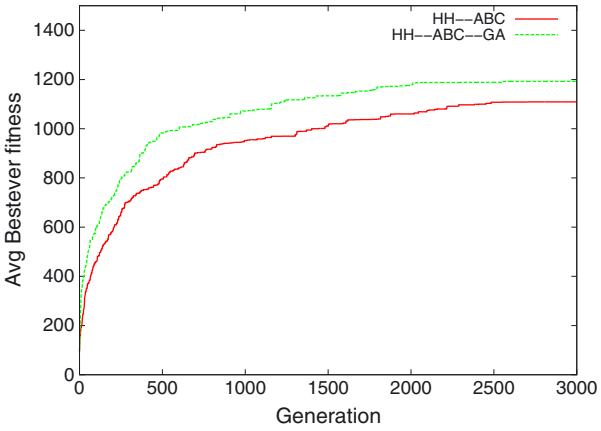

Figure 4. Performance of HH–ABC and HH–ABC-GA.

each generation. It is observed in this figure that the HH–ABC-GA leads to better individuals when compared with an HH–ABC, achieving, in this case, a gain that exceeds 7%. These results suggest that the hybridization of GA is advantageous for the ABC.

6.2. *Benchmark results and discussion*

In our experiments, four synthetic 27 amino acids long sequences were used as benchmark, as shown in the columns of the first row of the Tables I and II. These sequences have been used by other researchers for the 3DHP model [33, 34], and to the best of our knowledge, these sequences were used for the first time by [23] specifically for the 3DHP-SC model.

Both HH–ABC and HH–ABC–GA were run with 30 slaves per island. Results are shown in Tables I and II. The performance of each approach takes into account the best solution found and

Table I. Statistical results for benchmark sequences S_1 and S_2.

Sequence	S_1 $(PH)^3H^2P^2(HP)^2P^{10}H^2P$		S_2 $PH^2P^{10}H^2P^2H^2P^2HP^2HPH$	
Model	*Fitness*	T_r	*Fitness*	T_r
HH-ABC [11]	581.83 ± 35.25	1	673.95 ± 27.42	1
HH-ABC–GA	790.13 ± 86.03	1.026	900.95 ± 128.06	1.009

DOI: 10.1002/cpe

Table II. Statistical results for benchmark sequences S_3 and S_4.

Sequence	S_3 $H^4P^5HP^5H^3P^8H$		S_4 $H^3P^2H^4P^3(HP)^2PH^2P^2HP^3H^2$	
Model	*Fitness*	T_r	*Fitness*	T_r
HH-ABC [11]	772.36 ± 20.22	1	1002.59 ± 22.21	1
HH-ABC–GA	866.61 ± 73.64	1.08	1201.27 ± 166.19	1.029

the processing time. In this table, columns "*Fitness*" and "T_r" identify the average "*Fitness*" of the best solutions and the relative processing time, respectively.

Comparing the HH-ABC–GA and the HH-ABC, it is noticed that the quality of solutions obtained by the HH-ABC–GA was better as a result of the coevolution effect, thanks to the periodic migrations between independent subpopulations. It is also observable that the HH-ABC–GA presents a slight increase of the computational cost because of the improvement strategy (8% in the worst case).

7. CONCLUSIONS AND FUTURE WORK

The PSP is still an open problem in bioinformatics. Therefore, an important contribution of this work are the results regarding this issue. Because of the high computational cost necessary to tackle with the problem, even for small instances, the use of parallel/distributed computing was not only justified but also essential.

We investigated the application of ABC and GA in a hybrid-hierarchical concurrent model (HH-ABC-GA) for the PSP. In order to analyze the coevolutionary influence of one technique over another, each island of the concurrent hierarchical model is composed of a different algorithm in a ring topology, interleaving ABC and GA. Results showed that this combination took advantage of both the coevolution effect between concurrent evolutionary approaches and the different population-based search strategies.

Future work will focus on different topologies [6], a deep study of the migration policy between islands and local search strategies to improve the quality of the exploration of the search space, as well as to avoid stagnation in local maxima. Certainly, such improvements will require more computational power. Because the experiments were performed using a cluster computing environment, the communication overhead takes significant influence upon the computational time. Consequently, an appropriate load balance procedure has to be done. To overcome such drawback, future work will consider the use of alternative computing technologies, such as reconfigurable computing and General-Purpose Graphics Processing Units, to accelerate processing.

Hybridism of EC approaches is a recurrent issue in the literature. Here, we investigate a simple hybrid model with only two distinct algorithms. Future work will also investigate parallel hybrid versions with other swarm intelligence approaches, such as ACO [35], PSO [36], and firefly algorithm [37], so as to compare with the parallel ABC-GA presented in this study.

ACKNOWLEDGEMENTS

Authors would like to thank the Brazilian National Research Council (CNPq) for the research grant no. 305669/2010-9 to H. S. Lopes and C. M. V. Benitez, as well as to UDESC (Santa Catarina State University) and FUMDES program for the financial support to R. S. Parpinelli.

REFERENCES

1. Fogel DB. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3^{rd} edn. IEEE Press: Piscataway, NJ, 2006.
2. Poli R, Kennedy J, *et al*. Particle swarm optimization: an overview. *Swarm Intelligence* 2007; **1**(1):33–57.
3. Dorigo M, Stutzle T. *Ant Colony Optimization*. MIT Press: Cambridge, MA, 2004.

DOI: 10.1002/cpe

4. Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation* 2009; **214**:108–132.
5. Parpinelli RS, Lopes HS. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation* 2011; **3**(1):1–16.
6. Tavares LG, Lopes HS, *et al*. A study of topology in insular parallel genetic algorithms. *Proceedings of World Congress on Nature and Biologically Inspired Computing*, 2009; 632–635.
7. Venter G, Sobieszczanski-Sobieski J. A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. 6^{th} *World Congress of Structural and Multidisciplinary Optimization*, 2005; [sp].
8. Stützle T. Parallelization strategies for ant colony optimization. *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, 1998; 722–731.
9. Tirumalai V, Ricks KG, *et al*. Using parallelization and hardware concurrency to improve the performance of a genetic algorithm. *Concurrency and Computation: Practice and Experience* 2007; **19**:443–462.
10. Talbi EG. A taxonomy of hybrid metaheuristics. *Journal of Heuristics* September 2002; **8**:541–564.
11. Benítez CMV, Lopes HS. Parallel artificial bee colony algorithm approaches for protein structure prediction using the 3DHP-SC model, Intelligent Distributed Computing IV, 2010; 255–264.
12. Karaboga D. An idea based on honey bee swarm for numerical optimization, Engineering Faculty, Computer Engineering Department, Erciyes University, 2005.
13. Karaboga D, Akay B. Artificial bee colony (ABC), harmony search and bees algorithms on numerical optimization. *Proceedings of Innovative Production Machines and Systems Virtual Conference*, 2009; [sp].
14. Holland JH. *Adaptation in natural and artificial systems*. MIT Press: Cambridge, MA, USA, 1975.
15. Goldberg DE. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley: Boston, USA, 1989.
16. Nelson DL, Cox MM. *Lehninger Principles of Biochemistry*, 5^{th} edn. W.H. Freeman: Boston, USA, 2008.
17. Dill KA, Bromberg S, *et al*. Principles of protein folding – a perspective from simple exact models. *Protein Science* 1995; **4**(4):561–602.
18. Berger B, Leighton FT. Protein folding in the hydrophobic-hydrophilic HP model is NP-complete. *Journal of Computational Biology* 1998; **5**(1):27–40.
19. Lopes HS. Evolutionary algorithms for the protein folding problem: a review and current trends, Computational Intelligence in Biomedicine and Bioinformatics, 2008; 297–315.
20. Li MS, Klimov DK, *et al*. Folding in lattice models with side chains. *Computer Physics Communications* 2002; **147**(1):625–628.
21. Benítez CMV, Lopes HS. Hierarchical parallel genetic algorithm applied to the three-dimensional HP side-chain protein folding problem. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2010; 2669–2676.
22. Anfinsen CB. Principles that govern the folding of protein chains. *Science* 1973; **181**(96):223–230.
23. Benítez CMV, Lopes HS. A parallel genetic algorithm for protein folding prediction using the 3DHP side-chain model. *Proceedings of the IEEE Congress on Evolutionary Computation*, 2009; 1297–1304.
24. Lopes HS, Scapin MP. An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. *Lecture Notes in Computer Science* 2005; **3871**:238–246.
25. Beer Jr. F, Johnston ER, *et al. Vector Mechanics for Engineers – Statics*, 9^{th} edn. McGraw-Hill: New York, 2009.
26. Meysenburg MM, Foster JA. The quality of pseudo-random number generations and simple genetic algorithm performance. *Proceedings of the* 7^{th} *International Conference on Genetic Algorithms*, 1997; 276–282.
27. Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation* 1998; **8**(1):3–30.
28. Hembecker F, Lopes HS, *et al*. Particle swarm optimization for the multidimensional knapsack problem. *Lecture Notes in Computer Science* 2007; **4331**:358–365.
29. Scapin MP, Lopes HS. A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model, Success in Evolutionary Computation, 2007; 205–224.
30. Benítez CMV, Lopes HS. Protein structure prediction with the 3D-HP side-chain model using a master-slave parallel genetic algorithm. *Journal of the Brazilian Computer Society* 2010; **16**(1):69–78.
31. Sterling T. *Beowulf Cluster Computing with Linux*. MIT Press: Cambridge, MA, USA, 2002.
32. Gropp W, Lusk E, Thakur R. *Using mpi-2: Advanced Features of the Message-Passing Interface*. MIT Press: Cambridge, MA, USA, 1999.
33. Unger R, Moult J. Finding the lowest free energy conformation of a protein is an NP-hard problem: proof and implications. *Bulletin of Mathematical Biology* 1993; **55**:1183–1198.
34. Guo Y-Z, Feng E-M. The simulation of the three-dimensional lattice hydrophobic-polar protein folding. *Journal of Chemical Physics* 2006; **125**:234703.
35. Dorigo M, Stützle T. *Ant Colony Optimization*. MIT Press: Cambridge, MA, USA, 2004.
36. Kennedy J, Eberhart R. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*, 1995; 1942–1948.
37. Yang X-S. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press: York, UK, 2008.

DOI: 10.1002/cpe