J Braz Comput Soc (2010) 16: 69–78
DOI 10.1007/s13173-010-0002-6

ORIGINAL PAPER

Protein structure prediction with the 3D-HP side-chain model using a master–slave parallel genetic algorithm

César Manuel Vargas Benítez · Heitor Silvério Lopes

Received: 29 September 2009 / Accepted: 23 February 2010 / Published online: 24 April 2010

Abstract This work presents a master-slave parallel genetic algorithm for the protein folding problem, using the 3D-HP side-chain model (3D-HP-SC). This model is sparsely studied in the literature, although more expressive than other lattice models. The fitness function proposed includes information not only about the free-energy of the conformation, but also compactness of the side-chains. Since there is no benchmark available to date for this model, a set of 15 sequences was used, based on a simpler model. Results show that the parallel GA achieved a good level of efficiency and obtained biologically coherent results, suggesting the adequacy of the methodology. Future work will include new biologically-inspired genetic operators and more experiments to create new benchmarks.

Keywords Genetic algorithm · Bioinformatics · Protein folding · 3D-HP-SC

1 Introduction

Proteins are polymers composed by a chain of amino acids (also called residues) 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 [24].

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 3-dimensional structure. This process is called protein folding. The specific shape to which the protein naturally folds is known as its native conformation. Proteins are essential to life and they have countless biological functions. The specific biological function of a protein depends on its 3-dimensional shape, which in turn, is a function of its primary structure (its linear sequence of amino acids). That is why it is so important to study how proteins fold. 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.

Better understanding the protein folding process can result in important medical advancements and development of new drugs. 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 its 3-dimensional structure known. For instance, the UniProtKB/TrEMBL [17] repository of protein sequences has currently around 10 million records (as in December/2009), and the Protein Data Bank (PDB) [7] has the structure of only 62,000 proteins. This fact

This paper is an extended version of a paper that appeared at ENIA 2009 (The Brazilian Meeting on Artificial Intelligence).

C.M.V. Benítez (✉) · H.S. Lopes
Laboratório de Bioinformática, CPGEI, Universidade Tecnológica Federal do Paraná, Av. 7 de setembro, 3165, 80230-901 Curitiba, PR, Brazil
e-mail: cbenitez@cpgei.ct.utfpr.edu.br

H.S. Lopes
e-mail: hslopes@utfpr.edu.br

is due to the cost and difficulty in unveiling the structure of proteins, from the biochemical point of view.

Computer science has an important role here, proposing models for studying the Protein Structure Prediction (PSP) problem [20]. Nowadays, the simulation of computational models that take into account all the atoms of a protein is frequently unfeasible, even with the most powerful computational resources. Consequently, several simplified models that abstract the protein structure have been proposed. Although such models are not realistic, they use some biochemical properties of amino acids, and its simulation can show some interesting characteristics of real proteins. This is an important motivation for developing computational methods for predicting the structure of these proteins. The simplest computational model for the PSP problem is known as the 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 [3, 6, 10]. This fact has motivated the development of many metaheuristic approaches for dealing with the problem. In this scenery, evolutionary computation methods and, in special, Genetic Algorithms (GA) have been proved not only adequate, but very efficient [11, 20, 26].

The objective of this work is to develop a parallel master-slave GA for solving benchmark instances of an extension of the HP model, the 3D-HP-SC (see the next section). This model increases the realism of the simulation, but at the expense of increasing the complexity of the problem.

Comparing the current work with our previous ones [4, 5], this version describes in details the experiments for adjusting the parameters of the GA and includes a decimation strategy (Sect. 3.5), as well as updated results (regarding previous publications) and new benchmarks.

2 The 3D-HP side-chain model (3D-HP-SC)

The Hydrophobic-Polar (HP) model proposed by Dill [13] 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, most hydrophobic amino acids tend to group themselves in the inner part of the protein, in such a way to get protected from the solvent by the polar amino acids that are positioned preferably outwards. Therefore, a hydrophobic core is usually formed in a folded protein, especially in globular proteins.

In the HP model, the folding 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 [20].

From the biological point of view, the expressiveness of the HP models is very poor. Therefore, the next steps for improving it could be increasing the number of degrees of freedom by using more complex lattices, using two types of monomers or else including a side bead to represent the side-chain (SC) of the amino acids [18]. Therefore, a protein is modeled by a common backbone and a side-chain, either Hydrophobic (H) or Polar (P). This representation defines the 3D-HP-SC model, as shown in Fig. 1. This figure represents a hypothetical polypeptide of seven amino acids long.

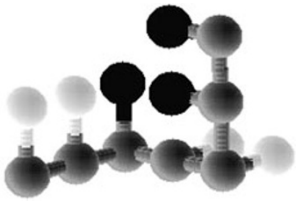

Fig. 1 Example of the 3DHP-SC model. *Black spheres* represent the hydrophobic side-chains (H), *light gray spheres* represent the polar side-chains (P), and *dark gray spheres* represent the backbone (the peptide bonds are also represented in *dark gray*)

In Fig. 1, it is possible to observe three hydrophobic side-chains close each other, forming nonlocal bonds. It is believed that the nonlocal hydrophobic side-chain bonds (HnC) are the main force that drives the protein folding process.

In the original HP model, it is considered that interactions between hydrophobic amino acids represent the most important contribution for the free-energy of the protein. The more hydrophobic interactions, the smaller the free-energy of the protein. For the 3D-HP-SC model, the free-energy of a given conformation is also in accordance with that principle, and takes into account the position that the side-chains occupy in the space. The free-energy of a conformation can be computed by Eq. 1 [18]:

$$H = \epsilon_{bb} \sum_{i=1, j>i+1}^{N} \delta_{r_{ij}^{bb}} + \epsilon_{bs} \sum_{i=1, j\neq i}^{N} \delta_{r_{ij}^{bs}} + \epsilon_{ss} \sum_{i=1, j>i}^{N} \delta_{r_{ij}^{ss}}. \quad (1)$$

In this equation, ϵ_{bb}, ϵ_{bs}, and ϵ_{ss} are the weights to the energy for each type of interaction: backbone/backbone (BB–BB), backbone/side-chain (BB–SC), and side-chain/side-chain (SC–SC); and r_{ij}^{bb}, r_{ij}^{bs}, and r_{ij}^{ss} are the distances (in the 3-dimensional space) between the ith and jth residues of interactions BB–BB, BB–SC, and SC–SC, respectively (for the sake of simplification, in this work, we used the unity distance between residues). Therefore, δ is an operator that returns 1 when the distance between the ith and jth side-chain is the unity, or 0 otherwise. As the amino acids chain folds over itself, bonds (or contacts) between

them take place, according to the possible interactions previously mentioned.

Side-chain beads and BB cannot occupy the same lattice site. This fact, known as collision, leads to an invalid conformation. During the folding process, the free energy of the protein tends to decrease. As mentioned before, the free-energy of a given 3-dimensional conformation is inversely proportional to the number of nonlocal hydrophobic side-chain bonds (*HnC*). Consequently, an algorithmic procedure for the PSP that maximizes *HnC* will, conversely, minimize the free-energy [13].

According to [18], the weight for *HnC* (ϵ_{HH}) is negative. Consequently, the smaller the value of the free-energy function, the closer to its native state the conformation will be, in accordance with the Anfinsen's thermodynamic hypothesis [1]. In this work, we consider the symmetric of H to turn the problem a maximization.

3 Methodology

Several approaches were proposed for the PSP problem. Many of them were based on evolutionary computation methods [20]. GAs have been shown very efficient for this problem, due to its simplicity and efficiency in finding good solutions for highly constrained problems in a complex search space (such as the PSP). Despite the effectiveness of GAs for studying the PSP problem, as the size of the protein to be analyzed increases, the computational effort can become unacceptable for a sequential version, and thus parallelized versions of GAs are more suitable than the ordinary sequential version.

In this work, a synchronous master-slave parallel GA [9] was developed. In this approach, the processing load is divided among several processors (*slaves*), under the coordination of a *master* processor. The master is responsible for initializing the population, performing the selection procedure, applying the genetic operators, and distributing individuals to slaves. Slaves, in turn, are responsible for computing the fitness function of each individual received. It will be shown later (Sect. 3.3) that the computation of the fitness function is very expensive, thus justifying the need for this type of parallel processing model. An analysis of performance of the master–slave GA was done elsewhere in [5].

3.1 Chromosome encoding

An important issue when using GAs for a given problem is the encoding of the chromosome that represents a possible solution to the problem. The encoding may have a strong influence not only in the size of the search space, but also in the hardness of the problem, due to the establishment of uncertain epistasis (influence of one gene into another one) between the genes of the chromosome.

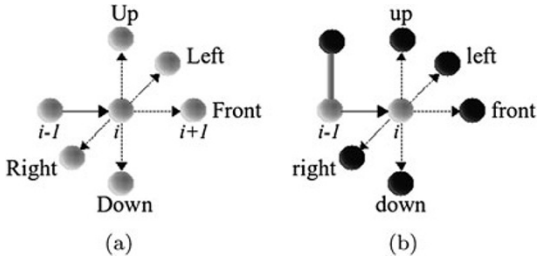

Fig. 2 Example of a relative movement of (**a**) backbone and (**b**) side-chain

There are several ways for representing a folding in a chromosome [20]: distance matrix, Cartesian coordinates (absolute coordinates), or relative coordinates. Krasnogor et al. [16] has studied the influence of the representation in the performance of GA using the 2D-HP model. They suggested that relative internal coordinates are the most efficient and, therefore, we used the same in our work. In this coordinates system, a given conformation of the protein is represented as a set of movements over a 3D cubic lattice. Thus, the position of each amino acid of the chain is described relatively to its predecessor.

As mentioned before, in the 3D-HP-SC model, the amino acids of the protein are represented by a backbone (B) and a side-chain, either hydrophobic (H) or polar (P). In the 3-dimensional space, there are five possible relative movements for the backbone (**L**eft, **F**ront, **R**ight, **D**own, **U**p), and other five for the side-chain, relative to the backbone (**l**eft, **f**ront, **r**ight, **d**own, **u**p). To illustrate movements in a cubic lattice, Figs. 2(a) and 2(b) show the possible movements for backbone and side-chain, respectively.

The combination of possible movements for backbone and side-chain gives 25 possibilities, represented by the following set: {Ll, Lf, Lr, Ld, Lu, Fl, Ff, Fr, Fd, Fu, Rl, Rf, Rr, Rd, Ru, Dl, Df, Dr, Dd, Du, Ul, Uf, Ur, Ud, Uu}. Each element of this set is translated to a unique symbol. Instead of the traditional binary alphabet, a set of 25 numbers and letters was used to encode the chromosome.

Considering the folding of a protein with n amino acids, a chromosome will represent the set of movements of its backbone and side-chain elements in the cubic lattice. Such chromosome will have $n - 1$ genes defined over the alphabet.

To represent the position of the amino acids in the cubic lattice, the Cartesian coordinates of each element (backbone and side-chain) will be later 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. Therefore, a progressive sequential procedure is necessary, starting from the first backbone, set to the origin of the coordinates system (point $(0, 0, 0)$), and its side-chain set at point $(0, -1, 0)$.

3.2 Initial population

The use of internal relative coordinates for the PSP leads to a problem when the initial population of the GA is created. Since individuals are randomly generated, the number of collisions between elements (backbone and side-chains) tends to be large [5, 20]. Consequently, in the generation of the initial population, there is no guarantee that valid individuals (without collisions) will be generated. In such situation, the GA will waste efforts evolving invalid conformations, before reasonable results can emerge. To overcome such condition, in this work, we propose a specialized method for generating the initial population.

The population is divided into two parts. Both are randomly generated, but one part is collision-free and the other is not. The rate of collision-free individuals in the initial population is a user adjusted parameter (in our experiments, it was fixed in 20% of *popsize*). The generation of collision-free individuals is done using a backtracking algorithm, explained below.

A folding is represented as an oriented 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 (backbone of side-chain) relative to its predecessor.

The backbone of the first amino acid is set to the origin, and its side-chain set to point $(0, -1, 0)$. The movement of the next amino acid backbone is randomly selected. If the movement leads to a collision with the backbone or the side-chain of any other amino acid previously positioned in the lattice, a backtracking is done in such a way to traverse the graph recursively, starting from the root (backbone of the first amino acid of the chain) and performing a depth-first search. At each node c, the algorithm verifies it this node is promising, that is, if the partial folding does not lead to collisions (either any elements—backbones or side-chains). Promising partial foldings can lead to a valid solution and then are maintained. If c is not promising, that is, one or more collisions were detected, the algorithm backtracks to the previous node of the graph and randomly selects another node not yet explored. The procedure is repeated until a complete valid solution is obtained. The backtracking algorithm is detailed in Algorithm 1.

Although the proposed method for generating the initial population is time-consuming, it assures the quality of individuals in the first generation, thus fostering the evolution of the GA toward good solutions.

It is important to note that the random number generator used in our GA was the Mersenne Twister [23], which is known as one of the best generators for this purpose.

Algorithm 1 Backtracking algorithm

```
1:  index ← 0
2:  c ← select_first_node()
3:  while solution s not complete do
4:     if promising(c) then
5:        s ← update_solution(c, index)
6:        c ← select_next_node()
7:        index ← index + 1
8:     else
9:        prune_subtree(c, index)
10:       if verify_neighborhood(index) then
11:          c ← reselect_node(c, index)
12:       else
13:          index ← index − 1 //backtracking
14:       end if
15:    end if
16: end while
```

3.3 Fitness function

Every individual, represented by a single chromosome is evaluated and its fitness value represents how good the solution is for the PSP problem in hand. The string of symbols encoded in the chromosome represents the spatial position of the amino acids of the chain, relative to their predecessors. This string is first decoded to another string of Cartesian coordinates and then used to compute the fitness function.

The fitness function used in this work is composed by terms that take into account not only the free-energy of the conformation, but also the number of collisions. The fitness function, shown in Eq. 2, also incorporates terms that measure the compactness of the hydrophobic and hydrophilic amino acids. This function was originally proposed by [21] for the 2D-HP model, and here adapted to the 3D-HP-SC model:

$$fitness = Energy \cdot RadiusG_H \cdot RadiusG_P \tag{2}$$

In this equation, the term *Energy* takes into account the number of nonlocal hydrophobic bonds (*HnC*), hydrophilic interactions, and interactions with the backbone. Also, the number of collisions (considered as penalties) and the penalty weight are considered in this term. $RadiusG_H$ and $RadiusG_P$ represent the gyration radius of the hydrophobic and hydrophilic side-chains, respectively. These terms are detailed below.

A given conformation under evaluation by the fitness function may have collisions, either between side-chains of different amino acids or between a side-chain and a backbone. Obviously, such conformation is physically invalid, but anyway the corresponding chromosome can carry some promising genetic material and should not be disposed. In

this case, there are three possible strategies to deal with this problem: simply discarding the invalid chromosome, fixing it or admitting it as a valid solution for the problem, but with a penalty. The first alternative is the easiest, but can throw away promising individuals and makes evolution very slow. The second one, although capable of avoiding invalid individuals, is computationally expensive. The third alternative is used in this work, that is, a penalty term is decremented from the free-energy term of Eq. 1. This penalty is composed by the number of points in the 3D lattice that is occupied by more than one element (*NC*—number of collisions), multiplied by the penalty weight (*PP*), as shown in Eq. 3. In all experiments, the penalty weight was empirically set to 10:

$$Energy = H - (NC \cdot PP). \tag{3}$$

A GA explores the search space of the problem both locally and globally. However, the shape of fitness landscape has a strong influence in the efficiency of the GA. In the case of the PSP problem, the fitness landscape is how the possible solutions (conformations) are distributed according to the hypersurface of the energy function. According to [13], under folding conditions, it is not the size of the energy landscape that counts, but its shape. The original HP model uses only the number of nonlocal hydrophobic interactions to evaluate individuals [13, 20]. This approach was demonstrated to have many plateaus in the fitness landscape [16], thus turning inefficient any local search method, due to the presence of many local maxima.

An indirect way to avoid the trap imposed by the complexity of fitness landscape to the GA was proposed by [21]. They used the physical concept of radius of gyration as part of the fitness function. Radius of gyration is a measure of compactness of a set of points (in this case, the side-chains of the amino acids in the lattice). The more compact the set of points, the smaller the radius of gyration. The radius of gyration is computed for two terms of the fitness function (Eq. 2), evaluating separately the sets of hydrophobic side-chains and polar side-chains. Equation 4 shows how this measure is computed:

$$RG_{aa} = \sqrt{\frac{\sum_{i=1}^{N_{aa}}[(x_i - \overline{X})^2 + (y_i - \overline{Y})^2 + (z_i - \overline{Z})^2]}{N_{aa}}}. \tag{4}$$

In this equation, x_i, y_i, and z_i are the coordinates of the ith side-chain of type "*aa*" of the protein, either hydrophobic (H) of polar (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 side-chains of type "aa."

In order to obtain a compact hydrophobic core, typical of globular proteins, the radius of gyration of the set of hydrophobic side-chains (RG_H) should be minimized, thus increasing the number of hydrophobic bonds between amino acids. Conversely, the maximization of the radius of gyration of the set of polar side-chains (RG_P) takes them to the outer side of the folding. To obtain such effect, both terms are computed by Eqs. 5 and 6, where $maxRG_H$ is the value of the radius of gyration when the amino acids chain is completely stretched. Once computed $RadiusG_H$ and $RadiusG_P$, they are used in the fitness function of Eq. 2:

$$RadiusG_H = maxRG_H - RG_H \tag{5}$$

$$RadiusG_P = \begin{cases} 1 & \text{if } (RG_P - RG_H \geq 0), \\ \frac{1}{1-(RG_P - RG_H)} & \text{otherwise.} \end{cases} \tag{6}$$

3.4 Selection method and genetic operators

The GA implemented in this work uses the k-tournament selection method, with $k = 3\%$ of *popsize*. This approach tends to be less elitist than other popular selection methods (such as the roulette wheel). After selection, individuals undergo the action of the genetic operators. In this work, we used only the standard genetic operators: two-point crossover and (multibit) mutation.

3.5 Improvement strategy: decimation-and-hot-boot

When a GA gets trapped around a local maxima in the search space, it is expected a decrement in the population diversity, mainly as a consequence of the crossover operator (local search). Sometimes, this effect can be counterbalanced by the action of the mutation operator. However, frequently, this is not enough and additional strategies are needed to avoid stagnation of the search.

When the population of the GA is concentrated around a local maxima, the only way to avoid useless computational effort is to escape from the current region, and to redirect the GA to the exploration of another regions of the search space. This is done using the Decimation-and-Hot-Boot (DHB) strategy [15, 25], explained below.

During the evolution of the GA, the best individual of each generation is always maintained (elitist strategy). An indirect evidence that the GA has stagnated is when the best individual does not improve for many generations. The strategy used verifies if the best-so-far individual is not improved from one generation to the next one. If so, a counter is incremented, otherwise, it is zeroed. When the counter reaches a predefined number of generations (*gen2decimate*), 50% of the population (randomly selected) is decimated and substituted by individuals generated according to the same procedure done for the initial population (see Sect. 3.2). It is important to note that the best individual is maintained during the DHB procedure.

The application of this strategy improves significantly the genetic diversity and allows the evolutionary process to continue for more generations. Ultimately, this improves the chances of finding even better solutions. However, it should be taken into account that new individuals recently created

by the DHB procedure probably will have low fitness values. Although the genetic diversity is improved, on the other hand the selective pressure is increased due to the large differences in fitness values of the individuals. It is well known that high selective pressure leads to premature convergence due to loss of genetic diversity. This is the opposite effect to what would be desired. Therefore, it is necessary to avoid high selective pressure during some generations just after the decimation. This is accomplished by decreasing the number of individuals that take part of the tournament selection (parameter *tourneysize*) to 2 during a fixed number of generations (parameter *gen2weakTourney*), and then returning to its original value.

4 Computational experiments

All experiments reported in this work were run in a cluster of 31 computers with the same hardware and software configurations (Intel *core 2-quad* at 3 GHz, running Linux). The application was developed in the ANSI-C programming language. To implement the message-passing interface between master and slave processes, we used the MPICH2 package [14], available in the internet at http://www.mcs.anl.gov/research/projects/mpich2/.

4.1 Benchmark sequences

In the experiments reported below, a total of 15 synthetic sequences were used, as shown in Table 1. These sequences have been frequently used by researchers to test the efficiency of algorithms for the PSP problem using 3D-HP models. It is important to note that, to date, there is no benchmark specifically designed for the 3D-HP-SC model.

Two groups of benchmarks were used: "Dill.*," first proposed by [28], and "Unger273D-*" by [27]. To the best of our knowledge, these sequences were used for the first time for the 3D-HP-SC model by [4] and [5], respectively. The first group of sequences have instances of 27, 31, and 36 amino acids, and the second group has sequences of the same size, 27 amino acids.

4.2 Processors' load balance

In a cluster-based parallel processing environment, the time needed to transmit messages between processes though the network is significantly high, when compared with the processors' speed. Hence, it is necessary to establish a suitable balance between the processing load of each processor and the amount of communication between processes.

Since the parallelization strategy used for the GA was the master-slave, the population size is evenly divided by the number of slave processors. Consequently, the population size indirectly determines the processing load for a given number of active processors.

Table 1 Benchmark sequences for the 3D-HP-SC model

Reference	n	*HP Chain*
Dill.1	27	$HP^4H^4P(PH)^3H(HP)^2PH^2P^2H$
Dill.2	27	$HP^3H^4(PH)^2HP^3HPH(HP)^2P^2HP$
Dill.3	27	$HPH^2(PPHH)^2H(HPPP)^2H^3P^2H$
Dill.4	31	$(HHP)^3H(HHHHHPP)^2H^7$
Dill.5	36	$PH(PPH)^{11}P$
Unger273d.1	27	$(PH)^3H^2P^2(HP)^2P^{10}H^2P$
Unger273d.2	27	$PH^2P^{10}H^2P^2H^2P^2HP^2HPH$
Unger273d.3	27	$H^4P^5HP^5H^3P^8H$
Unger273d.4	27	$H^3P^2H^4P^3(HP)^2PH^2P^2HP^3H^2$
Unger273d.5	27	$H^4P^4HPH^2P^3H^2P^{10}$
Unger273d.6	27	$HP^6HPH^3P^2H^2P^3HP^4HPH$
Unger273d.7	27	$HP^2HPH^2P^3HP^5HPH^2(PH)^3H$
Unger273d.8	27	$HP^{11}(HP)^2P^7HPH^2$
Unger273d.9	27	$P^7H^3P^3HPH^2P^3HP^2HP^3$
Unger273d.10	27	$P^5H(HP)^5(PHH)^2PHP^3$

Using a benchmark sequence, an experiment was performed to calibrate the processors' load. For a fixed number of generations, the population size (*popsize*) was changed between 200 and 2,000 individuals, and the number of processors was changed between 20 and 120. For each combination, 20 runs were done and the average computation time was recorded (Tp). Notice that computation time includes not only the processing time, but also the communication time between master and slaves.

We selected $popsize = 500$ and 100 slave processors (plus one master) because the balance between processing and communication achieves the best balance, that is, the smallest overall computation time. These values were used for all experiments.

4.3 Parameters adjustment

Many experiments were done to adjust all the parameters of the system, including the basic parameters of the GA (mutation and crossover probability, and tournament size) and parameters of the DHB strategy (see Sect. 3.5). The evaluation of these experiments takes into account not only the quality of the solutions obtained (according to the value of the fitness function), but also the computational effort spent (parallel computing time).

To decide which combination of parameters is better than others, we used the concept of Pareto optimality [12]. A plot is constructed in such a way to represent the behavior of two parameters, each of them to be minimized. Each point in the plot represents a possible combination of parameters. In our

case, the x axis is $[1 - normalized(AvgHnC)]$, a function of the average number of bonds between hydrophobic sidechains divided by the maximum value found. The y axis represents the average computing time (using the parallel master-slave cluster). Each point of the plot (x_i, y_i) is classified as dominated,when there is at least another point (x_j, y_j) such that $(x_j < x_i) \bigwedge (y_j < y_i)$, or nondominated, otherwise. In this analysis, only the nondominated points are really interesting, since they represent the best possible tradeoff between the two criteria. Notwithstanding, the Pareto plot also allows the user to find the most suitable working point for particular situations.

4.3.1 Running parameters of the GA

There is no specific procedure for adjusting running parameters of a GA for a given problem [19]. Although self-adjustment of parameters tends to be more efficient than hand-made adjustments [22], this is not the focus of the present work. Another strategy frequently used in the literature is setting a range for all important parameters of the GA and testing all possible combinations. This procedure is known as factorial experiment [8]. Practice has shown that the factorial experiment is satisfactory in most cases for adjusting the parameters of a GA.

The adjustment of parameters, in our case, is mainly focused on the quality of solutions, although the overall computation time is also an important factor to take into account. The factorial experiment was done using a typical benchmark sequence (see Sect. 4.1) of 27 amino acids. Both the number of generations (*maxgen*) and population size (*popsize*) were set to 3,000 and 500, respectively, for all experiments.

The basic parameters of the GA were tested in the following range: tournament size of the selection procedure (*tourneysize*): 2%, 3%, 5%; probability of crossover operator (*pcross*): 70%, 80%, 90%; probability of the mutation operator (*pmut*): 2%, 5%, 8%.

The combination of possible values for these parameters yields a total of 27 different experiments. Each experiment was run 30 times with different initial random seeds. Figure 3 presents a plot with the results obtained.

In Fig. 3, it is possible to observe that the difference between the two nondominated points (experiments 15 and 9) is less than 5%, regarding the computing time. Therefore, the former was elected due to better quality of solutions. The basic parameters of the GA corresponding to the experiment 15 are: $tourneysize = 3\%$, $pcross = 80\%$ and $pmut = 8\%$. These values were fixed for the remaining experiments.

4.3.2 Parameters of the DHB strategy

The DHB strategy explained in Sect. 3.5 has two parameters that affect the behavior of the GA. The first is named

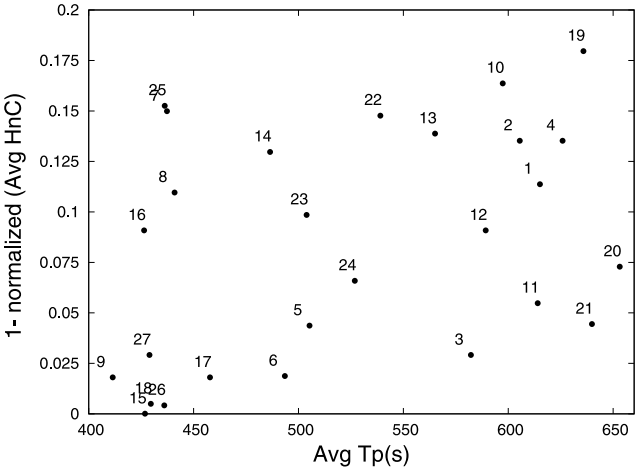

Fig. 3 Pareto plot for selecting the best set of basic parameters for the GA

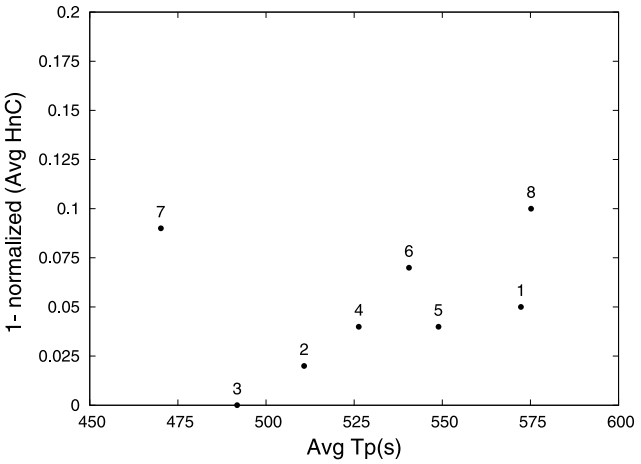

Fig. 4 Pareto plot for selecting the parameters of the DHB strategy

gen2decimate and identifies the number of generations without improvement of the best solution to start the DHB strategy. The other parameter is *gen2weakTourney* that represents the number of generations after decimation in which the value tourney size (*tourneysize*) is decreased. These two parameters were tested for the following values: [300; 600] and [30; 60; 300; 600], respectively. The combination of such values gives 8 experiments, which results are shown in Fig. 4. For each experiment, 30 independent runs were done with different initial random seeds.

In Fig. 4, the nondominated points correspond to experiments 3 and 7. The values of parameters corresponding to experiment 3 were chosen because they lead to solutions of better quality. These values are $gen2decimate = 300$ and $gen2weakTourney = 300$, and were set fixed for the remaining experiments.

The GA was executed with and without the DHB strategy for 100 independent runs. Figure 5(a) presents a plot of the average fitness of the best individual for the two sit-

uations above mentioned. It is observed in this figure that the use of the DHB strategy leads to better individuals when compared with a GA without DHB. The difference between curves starts around generation 500 when the DHB strategy is first applied. Figure 5(b) shows the average fitness of the best individual and the average of the average fitness (over the population at each generation). In this figure, we observe how the DHB strategy is efficient to preserve genetic diversity, since after its application, the distance from average to best is maintained (although slightly oscillating). Keeping high genetic diversity allows the GA to explore more efficiently the search space and avoids premature convergence.

5 Results and discussion

Due to the stochastic nature of GA, the parallel GA was run 100 times with different random seeds for each benchmark sequence. Results are shown in Table 2. In this table, the first column identifies the sequence, the second and third columns identify, respectively, the generation in which the best individual was found and the average number of generations needed to find the best individual (with the corresponding standard deviation). Next, the average processing time is reported considering the parallel processing in 100 slave processors. The last two columns of the table show, respectively, the average value (together with the corresponding standard deviation), and the maximum number of non-local bonds between hydrophobic side-chains.

In Table 2, it is observed that the parallel GA needed, on average, less generations than the maximum set ($maxgen = 3000$) to find the best-of-run solution. This fact suggests that

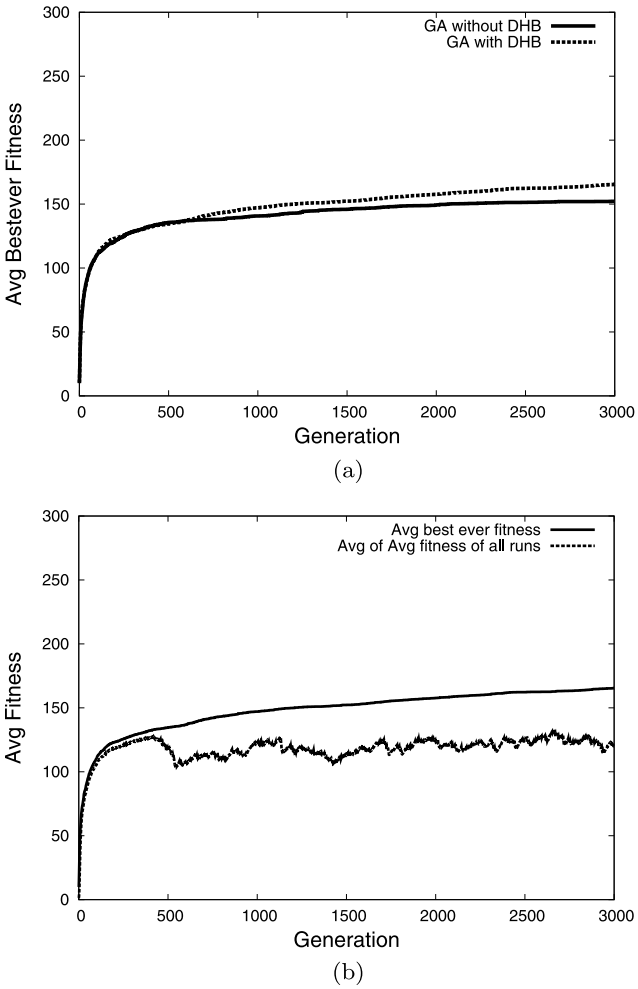

Fig. 5 Fitness plots

Table 2 Results for the benchmark sequences for the 3D-HP-SC model

Reference	Generation		avg T_p(s)	*HnC*	
	best	avg		avg	max
Dill.1	2136	1746.69 ± 873.65	373.83	15.07 ± 1.48	20
Dill.2	1806	1612.75 ± 875.62	378.88	11.98 ± 1.34	17
Dill.3	2764	1866.63 ± 926.63	375.21	14.88 ± 1.77	21
Dill.4	2407	2056.32 ± 725.76	394.66	28.98 ± 2.28	36
Dill.5	2282	1617.67 ± 843.62	414.99	11.18 ± 1.55	14
Unger273d.1	2979	1644.65 ± 876.51	474.93	8.36 ± 1.23	11
Unger273d.2	2514	1789.17 ± 875.38	475.79	8.73 ± 1.39	13
Unger273d.3	2251	1919.51 ± 790.87	466.01	9.12 ± 1.23	13
Unger273d.4	2979	2013.29 ± 776.86	479.62	15.13 ± 1.71	21
Unger273d.5	2089	1647.02 ± 944.61	465.31	8.77 ± 1.03	12
Unger273d.6	1974	1613.90 ± 897.92	461.22	9.24 ± 1.38	13
Unger273d.7	2323	1669.23 ± 931.98	473.88	10.60 ± 1.33	16
Unger273d.8	1094	1692.22 ± 970.30	474.37	3.90 ± 0.97	6
Unger273d.9	1802	1460.80 ± 876.65	481.97	6.63 ± 0.96	9
Unger273d.10	2207	1469.47 ± 936.20	470.33	9.60 ± 1.50	14

a smaller number of generations can be used in future experiments with these benchmarks. It is also observable that the higher the number of nonlocal bonds, the longer the GA takes to find the optimal solution. This fact suggests that the maximum achievable number of nonlocal bonds of a given folding may impose more difficulty to the problem than the length of the sequence itself. On the other hand, it is not surprising that the total processing time is directly dependent on the length of the sequence.

If the GA achieved the maximum number of nonlocal bonds (last column of Table 2) in all runs, its efficiency would be 100%. Considering all benchmark sequences, the proposed GA achieved an average efficiency that exceeds 70%. This value can considered good, if one takes into account factors such as the differences between instances, the stochastic nature of GAs, and number of parameters to be adjusted. We observed that the proposed GA performed consistently across runs and benchmarks. Therefore, results were not obtained by chance.

For some of the sequences tested, the best solution found corresponds to a folding that is shown graphically in Figs. 6(a), (b), (c), (d), (e), (f), and (g). In these figures, black spheres represent the hydrophobic side-chains (H), light gray spheres represent the polar side-chains (P), and dark gray spheres represent the backbone (the peptide bonds are also represented in dark gray).

It is possible to observe in these foldings that a compact hydrophobic core is formed, partially surrounded by amino acids with polar side-chains. This type of conformation, typical of globular proteins, was expected as a consequence of the fitness function. This behavior suggests that the proposed fitness function is adequate for the PSP problem using the 3D-HP-SC model, mainly because the final results are capable of mimicking some biological properties of real proteins during folding.

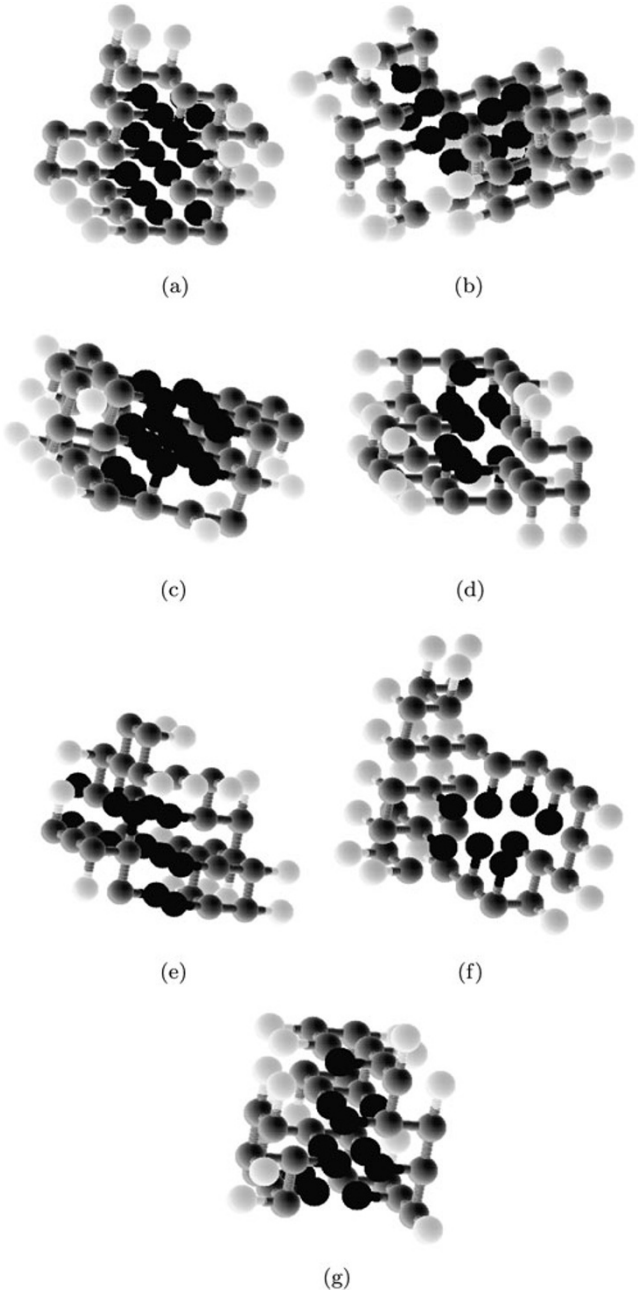

Fig. 6 Final 3D folding obtained for sequences Dill.3 (**a**), Dill.5 (**b**), Unger273d.4 (**c**), Unger273d.6 (**d**), Unger273d.7 (**e**), Unger273d.9 (**f**), and Unger273d.10 (**g**)

6 Conclusions and future work

The PSP is still an open problem for which there is no closed computational solution. Even the simplest discrete model for the problem requires an algorithm *NP*-complete, thus justifying the use of metaheuristic methods, such as genetic algorithms. While most works used the 3D-HP model, the 3D-HP-SC is still poorly explored, although being more expressive than the former, from the biological point of view.

The proposed parallel GA and its enhancements was able to find good solutions for the benchmark instances of the problem. The results shown in Table 2 and Fig. 6 are better (or, at least, equal) to other published works [4, 5], although it is not argued to be the optimal values. This fact suggests the adequacy of the proposed methodology. Notwithstanding, closely observing Fig. 6, it is possible to notice that small (but relevant) improvements could be done with those foldings. Such improvements could be achieved by means of local search algorithms hybridized with the GA. This issue will be addressed in future versions.

Due to the computational effort needed to cope with the problem, parallel processing was essential, allowing us to obtain good quality results in reasonable computing time. As the length of sequences increase, the demand for computational resources will increase accordingly. Therefore, future

work will also consider the use of reconfigurable hardware accelerators [2].

Overall results are good and very promising to suggest the continuity of the work. We believe that this work provides a contribution to this area of research because of three factors: exploring the 3D-HP-SC model, providing benchmark results useful for comparison with other approaches, and modeling an efficient genetic algorithm for the PSP problem. Future work will include the proposition of more benchmark instances, biologically-inspired genetic operators, and other parallelization strategies.

Acknowledgements The authors would like to thank the Brazilian National Research Council—CNPq, for the financial support under grants 550977/2007-4, 481207/2007-0, and 309262/2007-0 to H.S. Lopes and a M.Sc. scholarship to C.M.V. Benítez.

References

1. Anfinsen CB Principles that govern the folding of protein chains. Science (181)
2. Armstrong NB Jr, Lopes HS, Lima CRE (2007) Reconfigurable computing for accelerating protein folding simulations. Lect Notes Comput Sci 4419:314–325
3. Atkins J, Hart WE (1999) On the intractability of protein folding with a finite alphabet. Algorithmica 25(2–3):279–294
4. Benítez CMV, Lopes HS (2009) Algoritmo genético aplicado à predição da estrutura de proteínas utilizando o modelo 3D-HP side chain. In: Anais do VII encontro nacional de inteligência artificial (ENIA)
5. Benítez CMV, Lopes HS (2009) A parallel genetic algorithm for protein folding prediction using the 3DHP side-chain model. In: Proceedings of IEEE congress on evolutionary computation. IEEE Computer Society, Piscataway, pp 1297–1304
6. Berger B, Leighton FT (1998) Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. J Comput Biol 5(1):27–40
7. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) UniProt archive. Nucleic Acids Res 28(1):235–242
8. Box GE, Hunter WG, Hunter JS (2005) Statistics for experimenters: design, innovation, and discovery, 2nd edn. Wiley, New York
9. Cantú-Paz E (2000) Efficient and accurate parallel genetic algorithms. Springer, New York
10. Crescenzi P, Goldman D, Papadimitriou C, Piccolboni A, Yannakakis M (1998) On the complexity of protein folding. J Comput Biol 5(3):423–465
11. Custódio FL, Barbosa HJC, Dardenne LE (2004) Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm. Genet Mol Biol 27(4):611–615
12. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
13. Dill KA, Bromberg S, Yue K, Fiebig KM, Yee DP, Thomas PD, Chan HS (1995) Principles of protein folding—a perspective from simple exact models. Protein Sci 4(4):561–602
14. Gropp W, Lusk E, Thakur R (1999) Using MPI2: advanced features of the message-passing interface. MIT Press, Cambridge
15. Hembecker F, Lopes HS, Godoy W Jr (2007) Particle swarm optimization for the multidimensional knapsack problem. Lect Notes Comput Sci 4331:358–365
16. Krasnogor N, Hart WE, Smith J, Pelta DA (1999) Protein structure prediction with evolutionary algorithms. In: Banzhaf D, Eiben G, Honovar J, Smith S (eds) Proceedings of the international genetic and evolutionary computation conference, San Mateo, CA, pp 1596–1601
17. Leinonen R, Diez FG, Binns D, Fleischmann W, Lopez R, Apweiler R (2004) UniProt archive. Bioinformatics 20(17):3236–3237
18. Li MS, Klimov DK, Thirumalai D (2002) Folding in lattice models with side chains. Comput Phys Commun 147(1):625–628
19. Lobo FG, Lima CF, Michalewicz Z (2007) Parameter setting in evolutionary algorithms. Springer, New York
20. Lopes HS (2008) Evolutionary algorithms for the protein folding problem: a review and current trends. In: Computational intelligence in biomedicine and bioinformatics, vol I. Springer, Heidelberg, pp 297–315
21. Lopes HS, Scapin MP (2005) An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. Lect Notes Comput Sci 3871:238–246
22. Maruo MH, Lopes HS, Delgado MRB (2005) Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. Lect Notes Comput Sci 3448:154–165
23. Matsumoto M, Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans Model Comput Simul 8(1):3–30
24. Nelson DL, Cox MM (2008) Lehninger principles of biochemistry, 5th edn. Freeman, New York
25. Scapin MP, Lopes HS (2007) A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model. In: Yang A, Shan Y, Bui LT (eds) Success in evolutionary computation. Studies in computational intelligence, vol 92. Springer, Heidelberg, pp 205–224
26. Song J, Cheng J, Zheng T, Mao J (2005) A novel genetic algorithm for hp model protein folding. In: Proceedings of 6th international conference on parallel and distributed computing applications and technologies. IEEE Computer Society, Washington, pp 935–937
27. Unger R, Moult J (1993) A genetic algorithm for 3D protein folding simulations. In: Proceedings of the 5th annual international conference on genetic algorithms, pp 581–588
28. Yue K, Dill KA (1993) Sequence-structure relationships in proteins and copolymers. Phys Rev E 48(3):2267–2278