A Parallel Genetic Algorithm for Protein Folding Prediction Using the 3D-HP Side Chain Model

César Manuel Vargas Benítez, Heitor Silvério Lopes
Bioinformatics Laboratory, Federal University of Technology – Paraná (UTFPR),
Av. 7 de setembro, 3165 80230-901, Curitiba (PR), Brazil
cbenitez@cpgei.ct.utfpr.edu.br, hslopes@pesquisador.cnpq.br

***Abstract*— This work presents a methodology for the application of a parallel genetic algorithm (PGA) to the problem of protein folding prediction, using the 3DHP-Side Chain model. This model is more realistic than the usual 3DHP model but, on the other hand, it is has a higher degree of complexity. Specific encoding and fitness function were proposed for this model, and running parameters were experimentally set for the standard master-slave PGA. The system was tested with a benchmark of synthetic sequences, obtaining good results. An analysis of performance of the parallel implementation was done, compared with the sequential version. Overall results suggest that the approach is efficient and promising.**

Keywords: Genetic Algorithm, Bioinformatics, Protein Folding, 3DHP-SC.

I. Introduction

A protein is a polymer composed by a chain of amino acids (sometimes called residues), joined together by means of peptide bonds. Two amino acids form a peptide bond when the carboxyl group of one molecule reacts with the amino group of the other, thus releasing a molecule of water.

When proteins are under formation in the ribosome, they fold into a three-dimensional conformation. This process is known as protein folding. The biological function of a protein depends of its three-dimensional structure which, in turn, depends on its primary structure, that is, its amino acids sequence. It is known that ill-formed proteins (due to wrong folding) are the origin of some important diseases, such as cystic fibrosis, Parkinson's disease and some types of cancer. Due to its great importance for Medicine and Biochemistry, many research has been done about proteins and, consequently, many information is available. Therefore, acquiring knowledge about the three-dimensional structure of proteins and, consequently, about its functionality, is an important issue, since such knowledge can be used in the development of new drugs with specific functionality.

Notwithstanding, despite the growing number of proteins already discovered, only a few portion of them have they three-dimensional structure unveiled. This is because the difficulty involved in this endeavor, from the biological and biochemical point of view. It is here that Computer Science plays an important role, developing computational models and solutions for the protein folding problem. Due to the computational complexity, computational models that take into account every atom of the protein macromolecule are not feasible. Consequently, in recent literature, several simplified models for protein folding have been proposed. By using some biochemical properties of amino acids, such models, although far from reality, can display some interesting features useful for observing the behavior of synthetic proteins.

The simplest model for the study of protein folding is known as the hydrophobic-polar (HP) model, either in bi- (2D) or three-dimensional (3D) versions. Even with this simple model, the computational approach for its solution was proved to NP-complete that is, there is no polynomial-time algorithm to solve it [2]. This fact emphasizes the necessity of using heuristic methods 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 [13].

II. The 3DHP Side-Chain Model

The Hydrophobic-Polar (HP) model was proposed by Dill [5] and it is the most simple abstraction of the protein folding problem. This 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, most Hydrophobic amino acids are buried inside the protein, and protected from the solvent by the Polar amino acids that stand preferably outwards. The HP model considers that the interactions between hydrophobic amino acids represent the most important contribution for the free-energy of the protein. The more hydrophobic interaction, the small the free-energy of the protein.

In the HP model, the folding of a protein is represented in a lattice, usually square (for the bi-dimensional model – 2DHP) or cubic (for the three-dimensional model – 3DHP). Both 2DHP and 3DHP models have been frequently explored in the recent literature [13].

A next step to simulate more realistic features of proteins such as a dense hydrophobic core packing is to include a side bead representing a side chain (SC) of the amino acids [12]. Therefore, a protein is represented by a backbone (common to all amino acids) and a side-chain,

978-1-4244-2959-2/09/$25.00

either Hydrophobic (H) or Polar (P). This representation defines the 3DHP-SC model, as shown in Figure 1. In this figure, black cubes represent the hydrophobic side-chains, white cubes represent the polar side-chains, and gray cubes represent the backbone. For this model, the free energy of a given conformation takes into account the position in the space of the side-chain, and can be described as follows [12]:

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

Where ϵ_{bb}, ϵ_{bs} and ϵ_{ss} are the weights of 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 three-dimensional space) between the i^{th} and j^{th} residues of interactions BB-BB, BB-SC and SC-SC, respectively (for the sake of simplification, in this work we used unity distance between residues).

As the amino acids chain folds over themselves, contacts (or bonds) between them take place, according the possible interactions previously mentioned. It is believed that the hydrophobic interactions are the main driving force that causes the macromolecule to fold correctly over itself. During the folding process, the free energy of the protein tends to decrease. It is known that the free-energy of a given 3D conformation is inversely proportional to the number of hydrophobic contacts (HnC). Therefore, any algorithmic procedure for the folding that maximizes the HnC will, conversely, take the molecule to the smallest possible free-energy state.

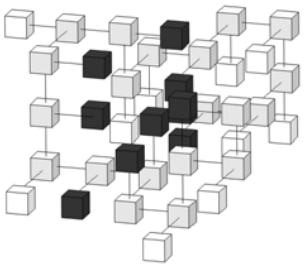


Fig. 1. Example of the 3DHP-SC model

III. Genetic Algorithms

Genetic Algorithms (GAs) are efficient search methods inspired in the natural evolution of living beings and based on the Darwinian model of natural selection. They have been developed by John Holland at the University of Michigan[9].

GAs operate on a population of individuals, which are an analogy with chromosomes. Each individual is a potential solution to a given problem and is typically encoded as a string (e.g., binary, character-based and real-valued encodings). Each chromosome consists of a set of "genes", with each gene being an instance of a particular "alleles" (e.g., 0 or 1).

After an initial population be randomly or heuristically generated, the algorithm evolves the population through sequential and iterative application of a selection procedure and two genetic operators: crossover and mutation. The selection procedure is applied to select individuals of the population to create an intermediate population. The genetic operations are applied to the selected individuals to create offsprings. Crossover exchanges subparts of two chromosomes, mimicking biological recombination between two single-chromosome organisms. Mutation randomly changes the allele values of some locations in the chromosome. At the end of each generation, a new set of individuals is created using pieces of the fittest individuals of the old population.

Although a lot of emphasis has been placed on the three above mentioned operators, there are two basic issues for applying a GA for a given optimization problem, because they are problem dependent: how the variables of the problem are encoded (coding scheme), and how the quality of a solution is measured (fitness function). The first issue is the representation problem, and the latter, the evaluation problem.

Amongst the many computational approaches for the Protein Folding Problem (PFP), certainly the most used is the genetic algorithm (GA), possibly due to its simplicity and efficiency in finding good solutions in large and complex search spaces. The ability of a GA in combining local features into a global solution makes it particularly appealing for the PFP [13].

GAs are generally able to find good solutions in reasonable amounts of time, but as they are applied to larger and harder problems there is an increase in the time requerid to find adequate solutions. As a consequence there have been multiple efforts to make GAs faster, and one of the most promising choices is to use parallel implementations [3].

IV. Implementation of the Parallel Genetic Algorithm

This section describes in detail the implementation of the parallel genetic algorithm applied to the PFP with the 3DHP-SC model.

Two versions of the genetic algorithm were developed: a sequential version and a synchronous master-slave parallel version. Both versions have exactly the same functionalities. In the sequential version everything is done in a single processor, while in the parallel version, the processing load is divided into 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

(crossover and mutation), and distributing individuals to slaves. Slaves, in turn, are responsible for computing the fitness function of each individual received. As will be shown in Section IV-C, the computation of the fitness function is very intensive, thus justifying the need for a parallel processing system. The software was developed in ANSI-C programming language, using the Message Passing Interface (MPI) for the communication between processes.

A. Encoding

An important issue when using GAs for a given problem is the encoding of the chromosome that represent a possible solution to the problem. The encoding can 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 between genes of the chromosome.

There are several ways for representing a folding in a chromosome [13]: distance matrix, Cartesian coordinates or internal coordinates. Extrapolating the study of [11] for the 2DHP model, we used in this work relative internal coordinates. 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.

In the 3DHP-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 three-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). Therefore, 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. This set of symbols, shown in Table I, is the alphabet used to encode the chromosome of the GA.

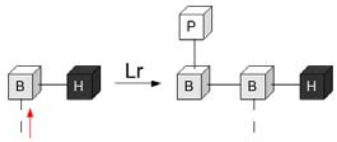
Considering the folding of a protein with n amino acids, a chromosome will represent the set of movements of its backbone and side-chain elements in the lattice. Such chromosome will have $n-1$ genes defined over the alphabet of Table I.

TABLE I

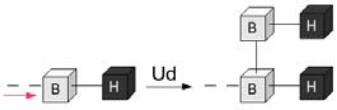
Encoding scheme of the relative internal coordinates.

Movements		Backbone				
		L	**F**	**R**	**D**	**U**
Side-chain	**l**	0	5	A	F	K
	f	1	6	B	G	L
	r	2	7	C	H	M
	d	3	8	D	I	N
	u	4	9	E	J	O

To illustrate the proposed encoding scheme, Figures 2a and 2b show two segments of sequences representing relative movements of both, backbone and side-chain: Lr (backbone to the left and side-chain to the right) and Ud (backbone upwards and side-chain downwards).



(a) Relative movement Lr



(b) Relative movement Ud

Fig. 2. Example of the relative movements Lr (a) and Ud (b)

To represent how the amino acids will be set in the 3D lattice, the Cartesian coordinates of each element (backbone and side-chain) will be later represented as x_i (row), y_i (column), z_i (depth), and obtained from the relative movement of current amino acid and position of its predecessor. Therefore, a recursive 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).

B. Initial Population

The use of internal relative coordinates for this problem leads to problem in the initialization of the GA, when the initial population is generated. Since it is randomly generated, the number of collisions between elements (backbone and side-chains) tends to grow [13], [16]. Consequently, in the generation of the initial population there is no guarantee that a valid individual will be generated. This leads the GA to spend a lot of effort evolving invalid conformations, before reasonable results can emerge. To overcame 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 a part is collision-free and the other may not. The rate of collision-free individuals in the initial population is a user adjusted parameter (see Table III). The generation of collision-free individuals is done using a backtracking strategy, as follows.

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 acids 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 and another movement is randomly chosen. In this case, the movement is selected from a set of possible movements without the one that caused the collision.

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

The random number generator used in our GA was the Mersenne Twister [15], which is known as one of the best generators for this purpose.

C. Objective Function

Every individual, represented by a single chromosome is evaluated and its fitness value represents how good the solution is. The string of symbols encoded in the chromosome represent the spatial position of an amino acid relative to its predecessor. This string is first decoded to another string of Cartesian coordinates and then used to compute the fitness function.

The fitness function proposed 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 Equation (2), also incorporates terms that measure the compactness of the hydrophobic and hydrophilic amino acids.

$$fitness = Energy \cdot RadiusH \cdot RadiusP \qquad (2)$$

Where: *Energy*: corresponds to the term that considers the number of hydrophobic contacts, hydrophilic interactions, and interactions with the backbone. Also, the number of collisions (considered as penalties) and the penalty weight; *RadiusH*: represents the gyration radius of the hydrophobic residues; *RadiusP* represents the gyration radius of the hydrophilic residues. These terms are detailed below.

1) Energy: The free-energy of a conformation with side-chain is shown in Equation (1). In this equation, the several possible interactions between residues and backbone are taken into account, that is backbone/backbone, backbone/side-chain, and side-chain/side-chain. With this equation it is also possible to give different weights when residues of a given interaction are hydrophobic or polar.

Recalling section II, the maximization of the hydrophobic contacts (HnC) is equivalent to the minimization of the free-energy of the conformation. Therefore, this problem is treated as a maximization problem, that is, the higher the number of HnC, the better. Therefore, the *Energy* term of the fitness function establishes a weight for this type of interaction that is larger than the weight for the remaining.

The conformation under evaluation can have collisions, either between side-chains of different amino acids or between a side-chain and a backbone. Such conformation is physically invalid, but anyway, it can convey some promising genetic material in the chromosome. There are three possible strategies to deal with this problem: simply discarding the invalid individual, fixing it or admitting it as a valid solution for the problem, but with a penalty. The first alternative can throw away promising individuals and make evolution very slow. The second one, although capable of avoiding invalid individuals, is computationally intensive. The third alternative is used in this work, that is, a penalty term is decremented from the *Energy* term of Equation (2). 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 Equation (3).

$$Energy = Energy - (NC * PP) \qquad (3)$$

2) RadiusH: An important issue regarding the prediction of the 3D structure of a protein (even using simple lattice models), is related to its energy landscape, that is, how the free-energy is distributed when considering all possible conformations of the protein. Following [6], under folding conditions, it is not the size of the energy landscape that counts, but its shape. However, it is multidimensional and, probably, have many local maxima.

The original HP model uses only the number of hydrophobic interactions to evaluate individuals [5], [13]. This approach was demonstrated to have many plateaus in the energy landscape [11], thus turning inefficient any local search method.

An indirect way to avoid the trap imposed by the energy landscape to the GA was proposed by [18]. 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 amino acids in the lattice). The more compact the set of points, the smaller the radius of gyration. Equation (4) shows how this measure is computed for the hydrophobic residues.

$$RgH = \sqrt{\frac{\sum_{i=1}^{N_H}(x_i - \overline{X}) + (y_i - \overline{Y}) + (z_i - \overline{Z})}{N_H}} \qquad (4)$$

Where x_i, y_i and z_i are the coordinates of the i-th hydrophobic residue of the protein; $\overline{X}$, $\overline{Y}$ and $\overline{Z}$ are the average of all x_i, y_i and z_i; and N_H is the number of hydrophobic residues of the protein.

RadiusH is computed according to Equation (5):

$$RadiusH = maxRgH - RgH \qquad (5)$$

3) RadiusP: This term of the objective function follows the same concept of *RadiusH*, previously mentioned.

$$RgP = \sqrt{\frac{\sum_{i=1}^{N_P}(x_i - \overline{X}) + (y_i - \overline{Y}) + (z_i - \overline{Z})}{N_P}} \qquad (6)$$

Where x_i, y_i and z_i are the coordinates of the i-th hydrophilic residue of the protein; $\overline{X}$, $\overline{Y}$ and $\overline{Z}$ are the average of all values of x_i, y_i and z_i; N_P is the number of hydrophilic residues of the protein.

$RadiusP$ is computed according to Equation (7):

$$RadiusP = \begin{cases} 1 & \text{if } (RgH - RgP \geq 0) \\ \frac{1}{1-(RgP-RgH)} & \text{otherwise} \end{cases} \quad (7)$$

D. GA Features

The GA implemented in this work uses the k-tournament selection method. 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.

Aiming at having an auxiliary method for controlling the selective pressure during evolution, the GA uses linear scaling of the fitness, as proposed by [7], and shown in Equation (8):

$$f' = af + b \quad (8)$$

Where: f' and f are the scaled fitness and original fitness, respectively; a and b are coefficients determined for each generation. Another way to express the relationship is using a parameter C_{mult} ($1.2 \leq C \leq 2.0$) that controls the span of fitness, as shown in Equation (9).

$$f'_{max} = C_{mult} * f_{avg} \quad (9)$$

By using the previous equation it is possible to establish a linear mapping in such a way that individuals with fitness around the population average will have delectable changes in their fitness. On the other hand, individuals with very high or very low values for fitness will have their values scaled down or up, respectively.

V. Experiments and Results

All experiments done in this work were run in a cluster of 30 networked computers. Each computer has an Intel Core 2 Duo processor at 2.66 GHz, 2 GBytes RAM. All computers run Linux and used MPICH2 [1], version 1.0 for the implementation of the message passing interface.

A. Benchmark

In our experiments, ten synthetic 27 amino acids-long sequences were used as benchmark, as shown in Table II. These sequences have been largely used by other researchers, for the 3DHP model [19], [16], [4], [8]. However, to the best of our knowledge, this is the first time they have been used for the 3DHP-SC model.

Only for comparison purposes, the maximum known number of hydrophobic contacts for folding of these sequences are shown in the column (E) of the table, following results obtained by [16].

[1]Available in: http://www.mcs.anl.gov/research/projects/mpich2/

TABLE II
Benchmarks for 3DHP model.

No.	*HP Chain*	E
273d.1	$(PH)^3H^2P^2(HP)^2P^{10}H^2P$	9
273d.2	$PH^2P^{10}H^2P^2H^2P^2HP^2HPH$	10
273d.3	$H^4P^5HP^5H^3P^8H$	8
273d.4	$H^3P^2H^4P^3(HP)^2PH^2P^2HP^3H^2$	15
273d.5	$H^4P^4HPH^2P^3H^2P^{10}$	8
273d.6	$HP^6HPH^3P^2H^2P^3HP^4HPH$	11
273d.7	$HP^2HPH^2P^3HP^5HPH^2(PH)^3H$	13
273d.8	$HP^{11}(HP)^2P^7HPH^2$	4
273d.9	$P^7H^3P^3HPH^2P^3HP^2HP^3$	7
273d.10	$P^5H(HP)^5(PHH)^2PHP^3$	11

B. Parameter Adjustment

There is no specific procedure for adjusting the running parameters of a GA for the problem dealt in this work. The experimental procedure done was to establish a range of values for the several running parameters and then try all possible combinations of them. For each combination of parameters, 10 independent runs with different initial random seeds were done. Other procedures for self-adjusting parameters of GAs were proposed elsewhere [14], but this issue falls outside the focus of the work. The best set of running parameters found with our factorial experiment is shown in Table III.

TABLE III
Best running parameters for the GA, obtained experimentally.

Parameter	Value
Number of generations	3000
Population size (*popsize*)	472
Collision-free rate	20% of *popsize*
Crossover rate	80%
Mutation rate	5%
Tourney size	3% of *popsize*
Linear scaling factor	$C = 1.4$

C. Performance Measures

1) Speedup: Probably, speedup is the most widely used performance measure in parallel computing [1]. This measure aims at evaluating how much a parallel algorithm is faster than the equivalent sequential version. Speedup (s_m) is defined as the time needed for running a given algorithm in one processor (T_1) divided by the running time of the same parallel algorithm, running in m processors (T_m), as in Equation (10).

$$s_m = \frac{T_1}{T_m} \quad (10)$$

From Equation (10) three types of speedup behavior can be clearly identified: sublinear speedup ($s_m < m$), linear speedup ($s_m = m$), and superlinear speedup

($s_m > m$). In [1], there is an interesting taxonomy for the measurement of speedup in parallel processing systems that was useful for this work:

- Strong speedup: compares the execution time of the parallel version with the best sequential version of the algorithm (also known as best-so-far sequential algorithm).
- Weak speedup: compares the execution time of the parallel version of an algorithm developed by a given programmer with its own sequential version of the algorithm. There are two variations for this sort of measure:
  - Versus panmixia: compares the parallelized algorithm with the sequential version;
  - Orthodox: compares the parallel algorithm running in one processor against the same algorithm running in m processors.

Since we do not know the best sequential genetic algorithm for protein structure prediction using the 3DHP-SC model, we cannot use the "Strong speedup" measure. Since we use the sequential version of the algorithm as a reference and the parallel (synchronous master-slave) with the same features to measure the speedup, we shall use the "Versus panmixia" approach to evaluate our implementation.

Figure 3 shows the speedup curve of the parallel implementation. In this figure it is observed that, although high, the speedup is always sublinear and tends to get apart from linear curve as the number of processors increase.

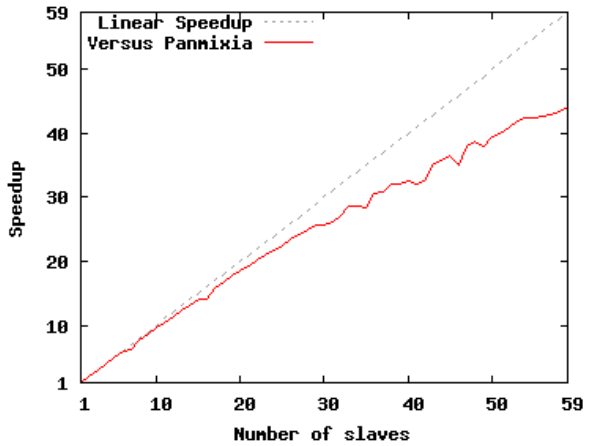


Fig. 3. Experimental speedup curve obtained for the implemented PGA running on a cluster

2) Efficiency: This measure is capable of evaluating the amount of time that a given processor is actually used for the processing [17]. Equation (11) shows how the Efficiency of a parallel computing system is computed: it is simply the acceleration factor divided by the number of processors.

$$e_m = \frac{s_m}{m} \tag{11}$$

Ideally, efficiency should be close to the unity (but not above it). However, in practice, this is not always possible, since processors are not used 100% of time for processing, but also for communication, memory allocation and other tasks of the underlying operating system [17]. Figure 4 shows the efficiency curve for our implementation. In this figure it is observed a moderated loss of efficiency as the number of processors increase. This is due to the communication overload caused in the master processor. Also, there are some situations in which a perfect load balancing between processors is not possible. This is due to the fact that the number of individuals to be processed divided by the number of processors does not give an exact number. This causes a loss of efficiency since, in a given instant of time, there will be some processors working and others waiting.

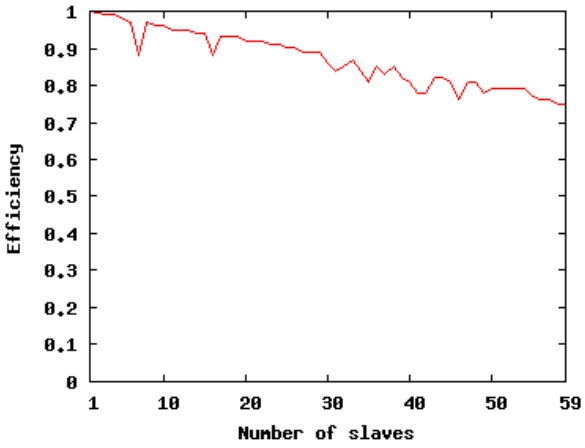
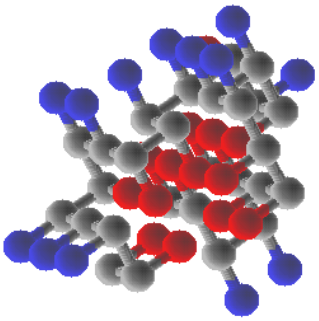


Fig. 4. Experimental efficiency curve obtained for the implemented PGA running on a cluster

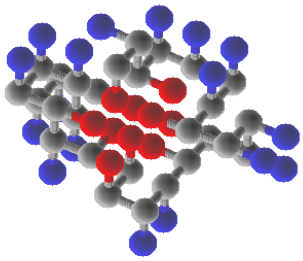
VI. Results

The proposed GA was applied to the benchmark problems described in Section V-A. For each sequence, 20 independent runs were done, with different initial random seeds. Results are shown in Table IV. In this table, the first column identifies the sequence, column "best" shows the number of the generation in which the best solution was found; columns "max" and "avg" show the maximum and average generations in which the best solution of the run was found; and T_p is the average processing time running in the cluster. Finally, the next three columns show, respectively, the maximum number of hydrophobic interactions found, the average value for all runs and the corresponding standard deviation.

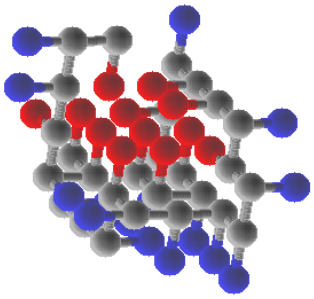
To illustrate the results obtained with our PGA approach, Figure 5 shows four examples of conformations found by the genetic algorithm.



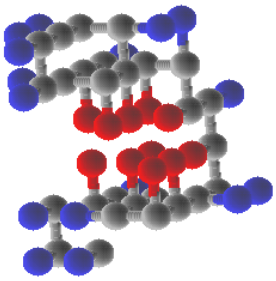
(a) Sequence 273d.4



(b) Sequence 273d.6



(c) Sequence 273d.7



(d) Sequence 273d.10

Fig. 5. Final 3D folding for sequences 273d.4 (a), 273d.6 (b), 273d.7 (c), 273d.10 (d). Blue balls represent the polar residues and Red ones represent the hydrophobic residues. The backbone and the connections between elements are shown in gray.

TABLE IV
RESULTS FOR THE BENCHMARK.

No.	# Generation				E		
	best	max	avg	T_p(s)	max	avg	σ
273d.1	1591	2811	1815.37	322.59	10	8.21	1.08
273d.2	1141	2985	1557.16	317.82	12	9.21	1.55
273d.3	2992	2992	1419.11	317.15	11	8.53	0.96
273d.4	1680	2800	1334.37	321.45	18	15.11	1.59
273d.5	2600	2939	1470.05	318.27	11	8.63	1.49
273d.6	692	2985	1517.63	317.31	13	10.26	1.45
273d.7	2420	2949	1809.57	320.63	16	11.86	2.28
273d.8	1127	2673	1620.05	315.87	6	4.36	0.83
273d.9	1589	2922	1360.37	316.91	9	6.4	1.17
273d.10	2638	2932	1837.05	322.26	14	10.16	1.83

VII. Conclusions and Future Work

This paper reports the first results of an ongoing project. Protein folding prediction is still an open problem in Bioinformatics. The 3DHP model has been frequently explored in the current literature. However, the 3DHP-SC, due to its higher level of complexity, is very sparcely approached, even more using evolutionary computation.

In this work we used a benchmark already used for the 3DHP model, but not yet for the 3DHP-SC. Therefore, an important contribution of this work are the results regarding this issue.

This work showed that a GA can be an efficient way do deal with the protein folding problem using the 3DHP-SC model. Although the results obtained cannot be considered optimal, they are coherent with the model, since that it is observed that the number of hydrophobic contacts found in the 3DHP-SC are always larger than in the 3DHP. To date, as mentioned before, these are the best results found for the 3DHP-SC model.

In Figures 5a, 5b, 5c and 5d it is observed the formation of a hydrophobic core. Due to the nature of the optimization problem, such core was already expected, suggesting that the proposed fitness function can capture some biochemical properties of the protein folding process. However, in those figures it can be observed that the hydrophobic core is partially protected by polar amino acids from the solvent. To mimic such behavior of real-world proteins another fitness function shall be devised in future work.

The use of parallel computing is justified in this work due to the necessary computational effort. For instance, if the system was run with a single computer, that is, a sequential algorithm, it would take around 5 hours each run. This shows how computationally-intensive the problem is, thus justifying the parallel approach. Future work will consider, also, hardware-based approaches, such as in [10], to accelerate processing.

Overall, results were good enough and promising to support the continuity of the work. We believe that this

work is an useful contribution to this area of research, since this model simulate more realistic features of proteins than the regular 3DHP. Further work will focus on more intensive experiments with these and other benchmarks, as well as the testing of other fitness functions and the development of knowledge-based genetic operators.

References

[1] H. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[2] J. Atkins and W.E. Hart. On the intractability of protein folding wit a finite alphabet of amino acids. *Algorithmics 25*, 25:279–294, 1999.

[3] Erick Cantú-paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, pages 611–615, 1998.

[4] F.L. Custódio, H.J.C. Barbosa, and L.E. Dardenne. Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm. *Genetics and Molecular Biology*, 27:611–615, 2004.

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

[6] K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas, and H.S. Chan. Principles of protein folding - a perspective from simple exact models. In *Protein Science*, 1995.

[7] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[8] Yu-Zhen Guo and En-Min Feng. The simulation of the three-dimensional lattice hydrophobic-polar protein folding. *The Journal of Chemical Physics*, 125:234703, 2006.

[9] J. H. Holland. *Adaptation in natural and artificial systems*. MIT Press, 1992.

[10] N.B. Armstrong Junior, H.S. Lopes, and C.R.E. Lima. Reconfigurable computing for accelerating protein folding simulations. *Lecture Notes in Computer Science*, 4419:314–325, 2007.

[11] N. Krasnogor, W.E. Hart, J. Smith, and D.A. Pelta. Protein structure prediction with evolutionary algorithms. In *International Genetic and Evolutionary Computation Conference (GECCO)*, pages 1596–1601, 1999.

[12] Mai Suan Li, D. K. Klimov, and D. Thirumalai. Folding in lattice models with side chains. *Computer Physics Communications*, 147(1):625–628, 2002.

[13] H.S. Lopes. Evolutionary algorithms for the protein folding problem: a review and current trends. In *Applications of Computational Intelligence in Bioinformatics and Biomedicine: Current Trends and Open Problems*, volume I, Heidelberg, 2008. Springer-Verlag.

[14] M.H. Maruo, H.S. Lopes, and M.R.B Delgado. Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. *Lecture Notes in Computer Science*, 3448:154–165, 2005.

[15] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[16] A.L. Patton, W.F. Punch III, and E.D. Goodman. A standard ga approach to native protein conformation prediction. *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995.

[17] S.H. Roosta. *Parallel Processing and Parallel Algorithms: Theory and Computation*. Springer-Verlag, New York, NJ, 1999.

[18] M.P. Scapin and H.S. Lopes. A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model. In *Success in Evolutionary Computation*, pages 205–224, Heidelberg, 2007. Springer-Verlag.

[19] R. Unger and J. Moult. A genetic algorithm for 3D protein folding simulations. *Proceedings of the Fitfth Annual International Conference on Genetic Algorithms*, pages 581–588, 1993.