An Improved Parallel Differential Evolution Approach for Protein Structure Prediction Using Both 2D and 3D Off-lattice Models

Diego H. Kalegari
Bioinformatics Laboratory
Federal University of Technology - Parana
Curitiba, Brazil
kalegari@gmail.com

Heitor S. Lopes
Bioinformatics Laboratory
Federal University of Technology - Parana
Curitiba, Brazil
hslopes@utfpr.edu.br

***Abstract*—Protein structure prediction (PSP) is a well-known problem in bioinformatics. Identifying protein native conformations makes it possible to predict its function within the organism. Knowing this also helps the development of new drugs and the comprehension of some biological mechanisms. During years some techniques have been developed for this purpose but, due to their high cost, it is necessary to use simplified models of protein structures. However, even the simplest models, with low biological plausibility, are excessively complex from the computational point of view. This paper reports the application of Differential Evolution (DE) to solve the PSP problem using a Toy Model (also known as the AB Model) in both 2D and 3D to represent the protein structure. This work presents two different versions of the DE algorithm (basic and adaptive) using a parallel architecture (master-slave) based on Message Passing Interface in a cluster. Some special operators for DE were developed: explosion and mirror mutation. All tests executed in this work used four benchmark sequences, ranging from 13 to 55 amino acids. The results for both parallel DE algorithms using both 2D and 3D models were compared with other works in the literature. The DE algorithm achieved excellent results. Overall results encourage further research towards the use of knowledge-based operators to improve further the performance of DE.**

I. Introduction

Scientists have tried to unveil and simulate the way proteins fold during their synthesis in the ribosome of the cell. Despite the huge efforts in this area, the folding of proteins to their functional structure is still an open issue. Therefore, the Protein Structure Prediction (PSP) has become a central problem in Molecular and Computational Biology. Current literature has established two different issues regarding PSP: first, how to describe the elements of a protein so that a convenient representation can be achieved, both from the biological and the computational point of view. Second, how to devise an effective method that find the state of minimum free energy of the protein, given a representation model.

Models that represent all the atoms of a protein are biologically plausible, but on the other hand, computationally unfeasible. Therefore, many researchers in Computational Biology have proposed alternative models to simplify the structure of a protein. Some of these models are based on a lattice representation and evaluates the free energy of the molecule to infer its state. For a comprehensive overview on computational models of protein structure and methods, see [1], [2].

The simplest models proposed for the protein structure optimization problem (both in 2D and 3D) are the HP lattice model [3], [4] and the AB off-lattice model [5], also known as Toy Model. In the HP model, the amino acids of a protein are converted into monomers H (hydrophobic) or P (polar or hydrophilic), depending on their affinity to water. Therefore a protein sequence is converted to a chain of monomers, and each monomer is placed in a point of a square or cubic lattice, in such a way their movements are constrained to the crosspoints of the lattice. Despite the simplicity of the HP model, exhaustive algorithms to solve the PSP using this model takes to a HP-hard computational problem [3].

The AB model [5] was inspired by the HP model. However, the main difference is that the position of monomers are not restricted to a lattice. They can be positioned anywhere in the plane (2D) or in the space (3D), connected by links (bonds) each other. This model is somewhat similar to the HP model since it translates the amino acids to two species of monomers: **A** (representing hydrophobic amino acids) and **B** (representing polar amino acids). This model uses a mathematical formulation to calculate the free energy of the conformation. The free energy is a central concept in understanding the properties of physical and chemical systems, including proteins.

In this work, we used DE [6], [7] to find the minimum energy of a conformation, also called the native conformation. DE is an evolutionary computation technique based in difference of vectors for generating perturbations in a vector population. DE was proven to be a flexible and efficient method or solving many interesting problems [8], [9], including bioinformatics [10], [11].

A. The AB Model

As mentioned before, the AB model considers two species of monomers, A (hydrophobic) and B (polar). In a chain, monomers are connected to each other by unity-length links and are spatially grouped in such a way that the links between them form angles, one torsion angle (θ) in 2D and the same

978-1-4673-5873-6/13/$31.00

torsion angle plus a rotation angle (ϕ) in 3D. Angles θ and phi are defined in the range $[-\pi, \pi]$, relative to the predecessor monomer. Consequently, a protein structure composed by n monomers represented using the AB model, has $n-2$ torsion angles, for 2D, and $n-2$ torsion angles plus $n-3$ rotation angles, for 3D.

The first mathematical formulation for the 2D-AB model was presented by [5]. Each monomer has a specific energy: **A** has an energy value of 1, and **B** has an energy value of -1. Considering two generic monomers i and j, of species ξ_i and ξ_j, the interaction between species give rise to different potential energy values (C_{2D}), as shown in Equation 1. For AA bonds the energy is 1, meaning that AA monomers tend to attract strongly to each other; BB bonds have energy $+1/2$, meaning that they have tendency of attracting to each other weakly; and AB or BA bonds have energy $-1/2$, meaning that when they are bonded they have tendency for weak repulsion.

$$C_{2D}(\xi_i, \xi_j) = \begin{cases} 1 & if \quad \xi_i, \xi_j = A \\ 1/2 & if \quad \xi_i, \xi_j = B \\ -1/2 & if \quad \xi_i \neq \xi_j \end{cases} \tag{1}$$

Therefore, considering r_{ij} the distance between the i^{th} and j^{th} monomers in the chain ($i < j$), the energy of a protein structure with n monomers (n-mers) in the 2D-AB model is given by Equation 2:

$$E_{2D} = \frac{1}{4}\sum_{k=1}^{N-2}(1 - cos\theta_k) + 4\sum_{i=1}^{N-2}\sum_{j=i+2}^{N}\left(\frac{1}{r_{ij}^{12}} - \frac{C_{2D}(\xi_i, \xi_j)}{r_{ij}^{6}}\right) \tag{2}$$

The above expression postulates two types of intermolecular potential energies. The first term depends only on the angles between monomers (torsion angle) and represents the backbone potentials; the second term represents the potential energy present in the non-bonded interactions, due to the interaction between monomers i and j, and it is known as the Lennard-Jones potential.

For the 3D model the formulation is somewhat similar to the 2D model, and was presented by [12] according to Equation 3.

$$\begin{aligned} E_{3D} \quad = \quad & -k_1\sum_{k=1}^{N-2}(\hat{b_k} \times \hat{b_{k+1}}) - k_2\sum_{k=1}^{N-3}(\hat{b_k} \times \hat{b_{k+2}}) \\ & +4\sum_{i=1}^{N-2}\sum_{j=i+2}^{N}\left(\frac{C_{3D}(\xi_i, \xi_j)}{r_{ij}^{12}} - \frac{C_{3D}(\xi_i, \xi_j)}{r_{ij}^{6}}\right) \end{aligned} \tag{3}$$

where:

$$C_{3D}(\xi_i, \xi_j) = \begin{cases} 1 & if \quad \xi_i, \xi_j = A \\ 1/2 & if \quad \xi_i, \xi_j = B \ \ or \ \ \xi_i \neq \xi_j \end{cases} \tag{4}$$

In this equation, $\hat{b_k}$ is the unit vector that represents the bond between monomers k and $k+1$. The vector product $(\hat{b_k} \times \hat{b_{k+1}})$ represents the energy from the interaction caused by the torsion angle. Constants $k_1 = -1$ and $k_2 = 1/2$ were empirically defined by [12], [13]. The second part the equation $(\hat{b_k} \times \hat{b_{k+2}})$ represents the energy caused by the rotation of monomers. The last term is the Lennard-Jones potential that represents the attraction forces present between the non-bonded monomers.

II. Methodology

A. Differential Evolution

DE is a metaheuristic optimization method from the evolutionary computation area and was proposed for solving polynomial fitting problems [6], [7]. The basic ideia is the use of difference of vectors for generating perturbations in a population of vectors. DE is conceptually simple, easy to implement and has proven to be flexible and efficient for hard optimization problems. The application of DE to real-world problems requests the tuning of some control parameters, as follows.

- **Population size** (*pop*): represents the number of candidate solutions (vectors) that the algorithm will handle at the same time.
- **Dimension of solutions** (*nDim*): defines the length of the vectors, in which each element is a variable of the problem.
- **Range of variables**: for each variable of the problem, its upper and lower bounds have to be defined.
- **Weighing factor** (*F*): (also known as differential weight) is a constant that multiplies the vector resulting from the difference between pairs of vectors (say, X_2 and X_3). *F* is a real-valued parameter in the range $[0, 1]$ (a typical value for *F* is 0.7).
- **Crossover probability** (*CR*): is the probability of crossing over a given vector of the population (X_i) and a vector created from the weighted difference of two vectors ($F.(X_2, X_3)$), that are applied to another vector [7]. It is also known as crossover rate [14]. This latter vector (X_i) can be either randomly chosen or the one with the best fitness found up to the moment (X_{best}). CR is defined in the range $[0, 1]$. There are two basic types of crossover: binomial (*Bin*), or exponential (*Exp*). The final result of the operation is a candidate vector ($X_{candidate}$). Although the value usually found in the literature for CR is 0.85, DE is very sensitive to this parameter and it is problem-dependent.
- **Strategy**: several different evolution strategies (vector operations) were proposed by [7], but the choice of such strategy is problem-dependent.
- **Stop criterion**: the time-out criterion is the most widely used, that is, the algorithm stops after a fixed number of iteration.

B. DE Vector Encoding

For applying DE to the PSP optimization problem focused in this work, individuals are encoded directly with the $n-2$ torsion angles for 2D, and $n-2$ torsion angles plus $n-3$ rotation angles for 3D, so as to represent the conformation of the n monomers of a given chain. Angles are defined based on the axis between the previous monomer and the current one, thus limiting values to the range $[-\pi, \pi]$, meaning that

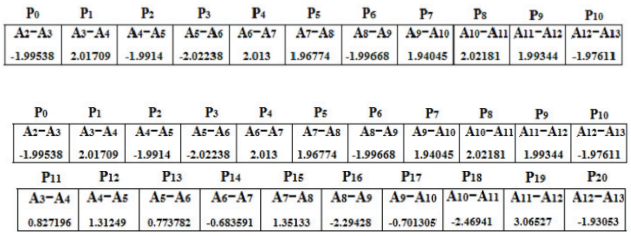

Fig. 1. Individual representation for a 2D sequence (top) and for a 3D sequence (botton).

a given angle is negative if the angle is below the axis, and positive otherwise. During the creation of the initial population angles are randomly generated within the predefined interval. Figure 1(top) represents a protein conformation with 13 monomers in 2D (with 11 torsion angles), and Figure 1(botton) represents the same conformation in 3D (with 11 torsion angles and 10 rotation angles).

C. Benchmarks

In order to compare the results and test the algorithm implemented we used a set of benchmark sequences found in the literature [15], [5]. The sequences were constructed using the Fibonacci series of monomers A's and B's are described in Table I, where N is the number of monomers in the sequence.

TABLE I
BENCHMARK SEQUENCES.

Sequence	N
ABBABBABABBAB	13
BABABBABABBABBABABBAB	21
ABBABBABABBABBABABBABBABBABABBAB	34
BABABBABABBABBA	
BABBABABBABBABABBABBABABBABABBABBABABBAB	55

D. Parallel Implementation

The evaluation of the fitness function (free energy) for the PSP problem is very costly. Therefore, a parallel version DE algorithm was implemented aiming at improving its performance. The parallel implementation was based on the MPICH-2[1], a portable and updated implementation of the message passing interface (MPI) [16]. MPI is a communication protocol widely used for parallel implementations of scientific applications. Basically, MPI provides a standardized means of communication and control between processes running in (the same or) different machines. By parallelizing a DE implementation using MPI, the computational load is divided and the overall performance is improved. A Master-Slave (MS) topology was proposed for the parallel DE, similarly as proposed by [17]. In this approach a Master process controls the DE algorithm and distributes to several Slave processes a number of individuals to be evaluated, that is, to compute the fitness function. As soon as a Slave process finishes its job, it reports the results to the Master and goes to an idle state until the Master allocates another job.

Master-slave tuning will define the number of slaves that will best fit to the problem. The number of slaves is closely linked to the number of individuals of the population. In order to achieve a good load balance it is necessary to observe two aspects. First, since the cluster is homogeneous, each slave needs to process the same number of individuals. Second, the processing time per slave has to be significantly higher than the time needed for transferring information from/to the master process.

The sequence of 34 monomers (Table I) has an average size among the benchmark, and it was used to tune the system. The same set of parameters was used to all configurations tested. Each configuration was tested for 50 independent runs with 100000 cycles, in the end the average processing time for each one was taken and expressed in the Table II as *Avg.Time*. In this table, Relative Time (μs) is the time to process each individual during the execution. DE sequential execution average time was 586 seconds.

TABLE II
MASTER-SLAVE TUNING TESTS.

#Indiv.	#Slaves	Individuals per slave	Avg.Time (s)	Relative Time (μs)
10	2	5	122.467	122.47
20	2	10	70.733	3.537
40	2	20	122.900	3.073
10	5	2	135.500	13.550
20	5	4	75.733	3.787
50	5	10	151.167	3.023
100	5	20	270.400	2.704
50	10	5	180.567	3.611
100	10	10	297.500	2.975
200	10	20	480.733	2.403
100	20	5	342.300	3.423
200	20	10	503.767	2.519
400	20	20	821.667	2.054
150	30	5	649.233	4.328
300	30	10	1028.967	3.430
900	30	30	2099.233	2.332
200	40	5	822.533	4.113
400	40	10	1286.167	3.215
100	1	1	586,00	586,00

Observing Table II, at the first sight considering only the Relative Time, one would choose the configuration of 400 individuals with 20 slaves as the best option. However, looking at the total time of the run, one can observe that it takes too much time to process. Based on both Time and Relative Time and aiming at minimizing resources usage, we selected the configuration with 100 individuals and 10 slaves, meaning that each slave will process 10 individuals at each cycle.

E. Parameter Tuning

The tuning of control parameters of evolutionary computation methods is usually problem-dependent. This is also the case of DE for parameters when considering not only the Strategy, but also F, CR. In the past, [6], [7] suggested that DE is not too sensitive to the control parameters, and they suggested as default values: strategy *Rand/1/Exp*, and

[1]http://www.mpich.org/

$CR = 0.85$. Indeed, most applications of DE to real-world problems use strategy and crossover value as constants. In order to check in what extent this assumption fits the PSP, a parameter tuning strategy was defined, as follows, with two sets of tests.

The first set of tests assumed $CR = 0.85$ and it was executed to find which value of F and which strategy most fit to the problem for all the benchmark sequences in both models (2D and 3D). The following strategies were tested: *rand/1/exp*, *randtobest/1/exp* and *best/1/exp*, and the following values for $F : 0.4, 0.6, 0.8$ and 0.95. For each sequence and for each model (2D and 3D) 50 independent runs were done using a population of 100 vectors during 100,000 cycles. Therefore, 96 tests were done. We found that for both, 2D and 3D, the best strategy was *rand/1/exp* (results not shown here). Regarding F, the most promising value was $F = 0.95$, although, for some instances 0.6 and 0.8 were also similar in quality of results, measured by the value of the energy (fitness function).

The second set of tests was done using *rand/1/exp* to verify which value of $F = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ is actually the best to be used. The remaining parameters and runs were set as before. Statistical tests were done by analyzing the variance with ANOVA, and the Tukey test. Results (not shown here), with $p < 0.05$ indicate that there are significant differences for the several values of F and, for values 0.8 and 0.9, there is no statistical difference. Since the experiments with $F = 0.8$ leaded to the best average, this value was chosen for further experiments with the 2D model. Similar tests and analysis were accomplished for the 3D model. In this case, the best performing value for F was 0.4 and this value was used henceforth.

III. Improvement Strategies

Three new strategies were implemented aiming at improving the performance of the basic DE algorithm. The strategies are explosion, mirror mutation, and adaptive DE, as described bellow.

A. Explosion

During the execution of the basic DE algorithm we observed the stagnation of the algorithm in some runs, especially in those with a large number of iterations. The explosion strategy, also known as decimation or mass extinction, is usually applied in swarm optimization methods [18]. It was implemented here to circumvent the premature convergence of the algorithm. In the DE algorithm, whenever an improvement of the best solution takes place, a no-improvement counter is reset, otherwise it is incremented. If this counter reaches a predefined value, stagnation has occurred, and the algorithm will restart the population, but keeping aside the best solution found so far as one of its individuals. Such approach is somewhat similar to other found in different metaheuristics. Different number of runs without improvement (*maxCount*) was tested with the same sequence and parameters of the previous experiment (F, CR and strategy). Other parameters were: 100 vectors, 100,000 cycles, and 100 independent runs. Results are shown in Table III and a similar statistical analysis as before was done.

TABLE III
Tuning tests of the explosion strategy.

	2D	3D
maxCount	average best fitness	average best fittness
1000	-7.64040	-56.9068
2000	-7.93238	-60.7432
3000	-8.28247	-59.9908
4000	-8.64264	-61.1384
5000	-8.73070	-62.4586
10000	-9.19448	-63.9156
15000	-9.43640	-64.8899
20000	-9.37592	-65.2869
25000	-9.23608	-63.5910
30000	-9.50384	-66.8663
35000	-9.16811	-66.2256
40000	-9.44631	-66.2664
45000	-9.17693	-66.9278
50000	-9.39401	-66.6370

As shown in the table, for both 2D and 3D, the best improvement in the results using explosion was achieved using 30000 iterations without evolution. Using this strategy, the average fitness obtained was 13% better than those obtained with the basic DE algorithm.

B. Mirror Mutation

Mirror mutation was created in order to include a local to the method, when the evolution is too slow (the fitness is still evolving, but improvements are at most at the 5$^{\text{th}}$ decimal). Similarly to the Explosion strategy, we used a counter to define the moment this strategy is performed (*maxMirror*). The local search works as follows: angles that represent the sequence will be mirrored, that is, if an angle is $-90°$, it will be changed to $90°$. Observe that this mirror also changes the positioning of the two monomers connected to the one that has this angle. Every time an angle is mirrored, the fitness function needs to be computed again, since the structure has changed. If the fitness of new structure just created is better than the previous one, it is kept, otherwise it is dismissed, and the previous structure is retrieved. The next angle of the sequence is tried and so on. Several values for the mirror mutation were tested using the 34 monomers sequence and the parameters previously defined. Results are presented in Table IV.

Figure 2 shows an example, for a 2D model, of what happens when an angle is mirrored. The figure at the top shows the monomers in their normal conformation, and the figure at the botton shows the monomers conformation after the selected angle is mirrored. The monomers sequence used on this example is ABABABAB, represented by the angles $[45, -90, 45, -90, 45]$, 'A' are represented by black dots and 'B' by white dots, the square dot represents the first monomer of the sequence.

Results shown in the table suggest that the mirror mutation strategy offers small improvement on results, 1% better, when compared the basic DE algorithm. When comparing with explosion strategy the results were 2% worst. Also, we

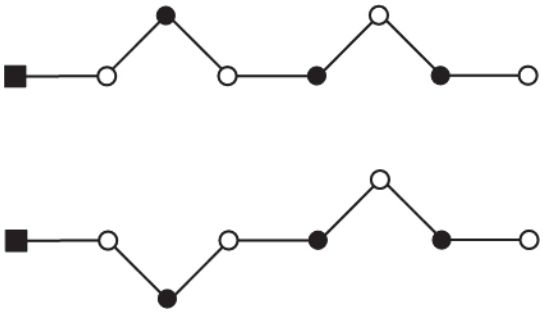

Fig. 2. Normal conformation (top) and mirror conformation (botton).

TABLE IV
TUNING TESTS OF THE MIRROR MUTATION STRATEGY.

maxMirror	average best fitness
5	-9.39130
10	-9.48385
20	-9.47538
30	-9.45525
40	-9.50386
50	-9.36287

observed that the number of slow improvement cycles to start this mutation does not affect significantly the result. Therefore, the value for the parameter was set to 40. For the 3D model, the mirror mutation is not detailed here due to its complexity.

C. Adaptive DE

Here we used a basic self-adaptive DE algorithm, as proposed by [19]. It was implemented aiming at improving the results obtained by the basic DE algorithm and the DE combined with explosion and mirror mutation strategies.

The self-adaptive DE implemented keeps fixed $CR = 0.85$ and the strategy $rand/1/exp$, but changes the actual F factor at each fitness evaluation, based on a simple normal distribution (*randN*) that is adapted according to the following rule:

$$\mu F = 1 - c.\mu F + c.Lmean(SF) \tag{5}$$

where μF is initialized with 0.5; c is a constant in the range $[0, 1]$ (usually, 0.5); $Lmean(SF)$ is the Lehmer mean [20], given by Equation 6; and F is defined as $F_j = randN(\mu F, 0, 1)$.

$$Lmean(SF) = \frac{\sum F_j^2}{\sum F_j} \tag{6}$$

The sequence of 34 monomers was used for testing this strategy, and the experiment was run 100 times, each time for 350,000 cycles. Results were better than those compared with the regular approach that uses F fixed. For this sequence the mean best fitness obtained was -9.42471 (for the 2D model) and -85.2752 (for the 3D model) which are slightly better than the basic DE for the same sequence.

IV. RESULTS AND DISCUSSION

A. Results for the 2D AB Model

The minimum energy for the four benchmark sequences based in the 2D model are shown in Table V. In this table, *DE*, *DE-EM*, *DE-Adp* and *DE-AdpEM* correspond, respectively, to the basic DE algorithm, DE with explosion and mirror mutation, adaptive DE and adaptive DE with explosion and mirror mutation. We also compared our best results with other found in the literature by other methods: the High Temperature Monte Carlo method (HTMC) [5] (currently accepted as the putative ground state energy), the pruned enriched Rosenbluth method (PERM) [15], the improved pruned enriched Rosenbluth method (PERM+) that uses subsequent conjugate gradient minimization [15], the Conformational Space Annealing (CSA) [21], and using a Particle Swarm Optimization method (PSO) [22]. Details about each if these algorithms is outside the scope of this work and can be found in the respective references. The comparison of results is shown in Table VI. It is worth to mention that statistical tests for comparing results cannot be done since mean values and standard deviation are not available for the other approaches found in the literature.

TABLE V
BEST RESULTS OBTAINED BY DE FOR THE 2D MODEL.

N	*DE*	*DE-EM*	*DE-Adp*	*DE-AdpEM*
13	**-3.1999**	**-3.1999**	**-3.1999**	**-3.1999**
21	**-6.1980**	**-6.1980**	**-6.1980**	**-6.1980**
34	-9.4237	-10.4699	-10.3822	**-10.5565**
55	-11.5240	-13.5205	-13.4719	**-17.3133**

TABLE VI
COMPARISON OF THE BEST RESULTS OF SEVERAL METHODS FOR THE 2D MODEL.

N	*HTMC*	*PERM*	*PERM+*	*CSA*	*PSO*
13	-3.2235	-3.2167	-3.2939	**-3.2941**	**-3.2941**
21	-5.2281	-5.7501	-6.1976	**-6.1980**	-6.1977
34	-8.9749	-9.2195	-10.7001	**-10.8060**	-10.7036
55	-14.4089	-14.9050	**-18.5154**	-17.9110	-18.4236

Results shown in the preceding tables show that the improved parallel DE has potential to reach the best minimum values for the four sequences. Results improved when explosion and mirror mutation were combined to the basic DE algorithm. Comparing the performance of the adaptive DE (*DE-Adp*) with basic DE algorithm, the results for the sequences with 13 and 21 monomers were exactly the same and for the those with 34 and 55 monomers a great improvement was observed, best fitness was 9% and 14% better, respectively. However, comparing *DE-Adp* with *DE-EM*, they have similar performance for the two smaller sequences, but *DE-EM* slightly outperforms *DE-Adp* for the larger sequences. Combining *DE-Adp* with mutation and explosion (*DE-AdpEM*) turned out to be one of the best approaches so far using the DE algorithm. Although it has the same performance of the basic DE for the two smaller sequences, for the sequences with 34 and 55 monomers the results were 10% and 33% better, respectively, then the basic DE, and 1% and 12% better then the *DE-EM*.

Comparing the best results obtained by the *DE-AdpEM* approach with the specialized methods that achieved the best results so far (*PERM+* and *CSA*), it turns out that our results

for the four sequences are: -2.85%, 0%, -2.36%, and -6.49%, respectively. Recalling that HTMC is considered the ground state energy by the creators of the benchmark, our approach overrid such values for the sequences of 21, 34 and 55 monomers.

The visual quality of the folding produced by the best run of the *DE-AdpEM* algorithm are shown in Figure 3(a) to 3(d). These figures shows the conformation of the synthetic protein folds in the plane. A MATLAB[2] program was developed to convert the string of angles into x, y into Cartesian coordinates and plot the structure. The square dot represents the start of the sequence, which can be either an A or B monomer; black dots represent A monomers and the white dots represent B monomers.

B. Results for the 3D AB Model

According to the previous experiments, the basic *DE* and the *DEAdp* were surpassed by the more improved *DE-EM* and *DE-AdpEM* methods. Therefore, the former were not used for the 3D model, and Table VII shows the best results found by the proposed *DE-EM* and *DE-AdpEM* methods.

TABLE VII
BEST RESULTS OBTAINED BY DE FOR THE 3D MODEL.

N	*DE-EM*	*DE-AdpEM*
13	-26.5066	**-26.507**
21	-48.9873	**-50.3613**
34	-89.7957	**-92.0962**
55	-149.5675	**-157.112**

Other results of specialized methods found in the literature for the 3D model are shown in Table VIII, as follows: *MUCA* is the multicanonical method [13]; *ELP* is the Energy Landscape Paving minimization [23]; *ACMC* is the Annealing Contour Monte Carlo algorithm [24] and *ACMC+* is the same algorithm with further improvement [24]; *CSA* is the Conformational Space Annealing [21]; and PSO is a Particle Swarm Optimization approach from [25].

TABLE VIII
COMPARISON OF THE BEST RESULTS OF SEVERAL METHODS FOR THE 3D MODEL.

N	*MUCA*	*ELP*	*ACMC*	*ACMC+*	*CSA*	*PSO*
13	-26.496	-26.498	-26.363	**-26.507**	-26.4714	-24.888
21	-52.915	**-52.917**	-50.860	-51.718	-52.7865	-46.611
34	-97.273	-97.261	-92.746	-94.043	**-97.7321**	-80.409
55	-169.654	-172.696	-149.481	-154.505	**-173.9803**	-115.758

Comparing the two versions of DE, *DE-AdpEM* has achieved improvements of 0%, 2.8%, 2.5% and 5% over *DE-EM* for the four benchmarks. This improvement is significant, considering the hardness of the 3D model compared with the previous 2D model. Furthermore, the difference increases in favor of *DE-AdpEM* as the size of the chain increases.

The comparison of *DE-AdpEM* with the other approaches in the literature (Table VIII) is more complex. For the smallest

[2]http:\\www.mathworks.com

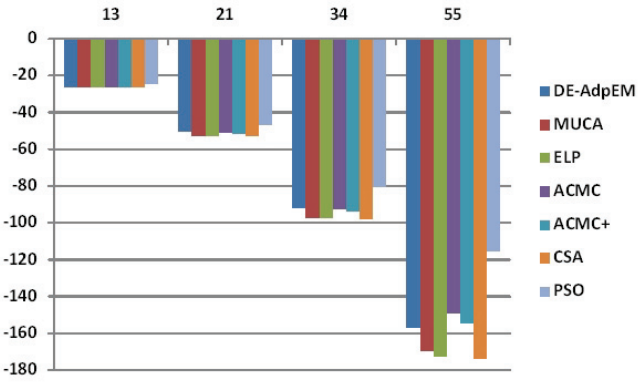

Fig. 4. Comparison of the best results achieved by all approaches for the 3D model.

sequence ($N = 13$) *DE-AdpEM* was better (or the same) than the remaining methods. For the intermediary sequences ($N = 21$ and $N = 34$), *DE-AdpEM* was around 5% worse than *MUCA*, *ELP* and *CSA*, and around 1∼2% worse than *ACMC* and *ACMC+*. These differences are not surprising, since all but *PSO* are specialized methods. Regarding *PSO*, *DE-AdpEM* was significantly better, outperforming the latter in 6.5%, 8%, 14.5% and 35.7% for the four benchmarks. Finally, considering only the best-of-all results of all approaches, *DE-AdpEM* was 0%, 4.8%, 5.7% and 9.7% worse, for the four benchmarks. A visual comparison of results for the 3D model can be seen in Figure 4.

V. CONCLUSIONS

The auto-adaptive approach of DE with enhanced strategies of explosion and mirror mutation was the version with best results. Overall, the results achieved by the improved DE proposed here suggest that this method is adequate and very promising for solving the PSP problem based on the off-lattice AB model. This is particularly true when comparing results with specialized methods in the literature, taking into account that DE is a general-purpose optimization method.

At this point, it is very important to note that many recent papers in the literature use variants of energy functions (specially for the 3D model), what precludes direct comparison of numerical results between methods. Furthermore, comparisons between different methods have to be done with care, since, most times the number of energy evaluations is not explicitly mentioned and, therefore, different computational efforts have been made to achieve the published energy levels. This is particularly true for the specialized methods cited before. Consequently, a fair comparison across different methods in uncontrolled situations is virtually impossible. Anyway, the proposed DE is competitive.

The parallelization of DE was essential to allow the simulations in acceptable processing time. Future work will address the use of reconfigurable computing as well as general-purpose graphics processing unity (GP-GPU) boards [26] as hardware accelerators. Although the impact of the mirror mutation was

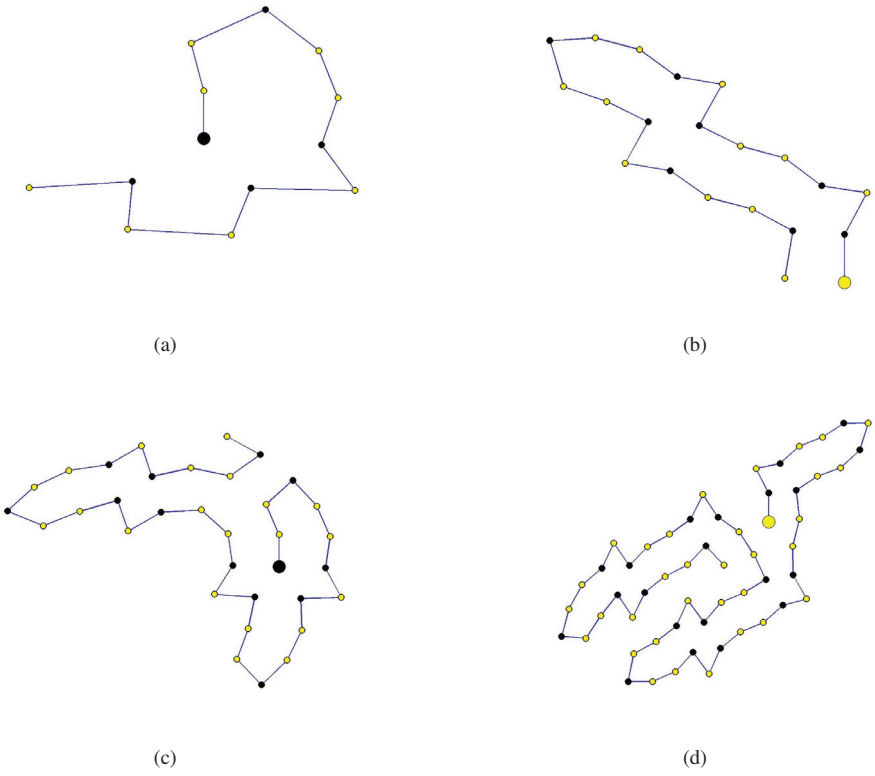

Fig. 3. Best results for *DE-AdpEM* for sequences: N=13 (a), N=21 (b), N=34 (c) and N=55 (d).

not significant, we believe that the use of special operators can take important improvements in the methods for the PSP, specially for the metaheuristics. In the same way, self-adaptation turns out to be an important resource not only for avoiding exhaustive fine-tuning of parameters, but also, for improving performance. In the near future we intend to develop more accurate methods for comparing systematically different approaches. Overall, the promising results reported here encourages further research.

ACKNOWLEDGMENT

This work was partially supported by the Brazilian National Research Council – CNPq, under Research Grant No. 305669/2010-9 H.S Lopes.

REFERENCES

[1] 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*, T. G. Smolinski, M. M. Milanova, and A.-E. Hassanien, Eds. New York, USA: Springer, 2008, pp. 297–315.

[2] K. Merz Jr. and S. LeGrand, Eds., *The Protein Folding Problem and Tertiary Structure Prediction*. Boston, USA: Birkhauser, 1994.

[3] J. T. Ngo, J. Marks, and M. Karplus, "Computational complexity, protein structure prediction and the Levinthal paradox," in *The Protein Folding Problem and Tertiary Structure Prediction*, K. Merz Jr. and S. LeGrand, Eds. Boston, USA: Birkhauser, 1994, pp. 433–506.

[4] 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 pespective from simple exact models," *Protein Science*, vol. 4, pp. 561–602, 1995.

[5] F. H. Stillinger, T. Head-Gordon, and C. L. Hirshfeld, "Toy model for protein folding," *Physical Review E*, vol. 48, pp. 1469–1477, 1993.

[6] R. M. Storn and K. V. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[7] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[8] M. A. Falcone, H. S. Lopes, and L. S. Coelho, "Supply chain optimisation using evolutionary algorithms," *International Journal of Computer Applications in Technology*, vol. 31, pp. 158–167, 2008.

[9] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis, "A review of major application areas of differential evolution," in *Advances in Differential Evolution*, U. K. Chakraborty, Ed. New York, USA: Springer, 2008, pp. 197–238.

[10] D. H. Kalegari and H. S. Lopes, "A differential evolution approach for protein structure optimisation using a 2D off-lattice model," *International Journal of Bio-Inspired Computation*, vol. 2, pp. 91–106, 2010.

[11] H. S. Lopes and R. A. Bitello, "A differential evolution approach for protein folding using a lattice model," *Journal of Computer Science and Technology*, vol. 22, pp. 904–908, 2007.

[12] A. Irbäck, C. Peterson, F. Potthast, and O. Sommelius, "Local interactions and protein folding: A 3D off-lattice approach," *Chemical Physics*, vol. 107, no. 1, pp. 273–282, 1997.

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

[14] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing*, vol. 9, no. 3, pp. 1126–1138, 2009.

[15] H. P. Hsu, V. Mehra, and P. Grassberger, "Structure optimization in an off-lattice protein model," *Physical Review E*, vol. 68, no. 3, p. 037703, 2003.

[16] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel*

Programming with the Message-Passing Interface. Cambridge, USA: MIT Press, 1999.
[17] C. M. V. Benítez and H. S. Lopes, "Protein structure prediction with the 3D-HP side-chain model using a master slave parallel genetic algorithm," *Journal of the Brazilian Computer Society*, vol. 16, pp. 69–78, 2010.
[18] F. Hembecker, H. S. Lopes, and W. Godoy Jr., "Particle swarm optimization for the multidimensional knapsack problem," *Lecture Notes in Computer Science*, vol. 4431, pp. 358–365, 2007.
[19] J. Zhang and A. C. Sanderson, *Adaptative Differential Evolution, A robust approach to Multimodal Problem Optimization*. Berlin, Germany: Springer-Verlag, 2009.
[20] P. S. Bullen, *Handbook of means and their inequalities*. Berlin, Germany: Springer-Verlag, 1987.
[21] S. Y. Kim, S. B. Lee, and J. Lee, "Structure optimization by conformational space annealing in an off-lattice protein model," *Physical Review E*, vol. 72, pp. 1–6, 2005.
[22] X. Zharg and T. Li, "Improved particle swarm optimization algorithm for 2D protein folding prediction," in *1st International Conference on Bioinformatics and Biomedical Engineering*, 2007, pp. 53–56.
[23] U. H. E. Hansmann and L. T. Wille, "Global optimization by energy landscape paving," *Physical Review Letters*, vol. 88, no. 6, p. 068105, 2002.
[24] F. Liang, "Annealing contour Monte Carlo algorithm for structure optimization in an off-lattice protein model," *Chemical Physics*, vol. 120, no. 14, pp. 6756–6763, 2004.
[25] R. S. Parpinelli, C. M. V. Benítez, J. Cordeiro, and H. S. Lopes, "Performance analysis of swarm intelligence algorithms for the 3D-AB off-lattice protein folding problem," *Journal of Computational and Theoretical Nanoscience*, vol. 10, 2013, to appear.
[26] M. H. Scalabrin, R. S. Parpinelli, C. M. V. Benítez, and H. S. Lopes, "Population-based harmony search using GPU applied to protein structure prediction," *Journal of Computational Science and Engineering*, vol. 8, 2013, to appear.