A differential evolution approach for protein structure optimisation using a 2D off-lattice model

Diego Humberto Kalegari* and Heitor Silvério Lopes

Bioinformatics Laboratory,
Federal University of Technology – Paraná,
Av. 7 de setembro, 3165 80230-901 Curitiba (PR), Brazil
E-mail: kalegari@gmail.com
E-mail: hslopes@pesquisador.cnpq.br
*Corresponding author

Abstract: Protein structure optimisation is a well-known problem in bioinformatics. This work applies an evolutionary algorithm to solve the protein structure optimisation problem based on the AB off-lattice model. Three different implementations of the differential evolution (DE) algorithm were developed, a sequential and two parallel. The parallel implementations (master-slave and ring-island) showed superior performance than the sequential one. Experiments were done using a benchmark of toy sequences with 13 to 55 monomers long. Results of the DE implementations were compared with other works in the literature. Good results were achieved for most sequences, not achieving the optimal values, but competitive with other specialised methods. Overall results encourage further research towards the use of knowledge-based operators to improve performance of DE.

Keywords: bioinformatics; protein folding; evolutionary computation; differential evolution; parallel computation; message passing interface; MPI.

Reference to this paper should be made as follows: Kalegari, D.H. and Lopes, H.S. (2010) 'A differential evolution approach for protein structure optimisation using a 2D off-lattice model', *Int. J. Bio-Inspired Computation*, Vol. 2, Nos. 3/4, pp.242–250.

Biographical notes: D.H. Kalegari received his BSc in Computer Engineering from Pontificial Catholic University of Paraná, in 2005. He is currently a Development Engineer at Institute of Technology for Development (Lactec). He is also an MSc candidate in Computer Science at the Federal University of Technology of Paraná (UTFPR), Brazil. His current research interests are bioinformatics, evolutionary computation, parallel computing and SOA.

H.S. Lopes is an Associate Professor at the Department of Electronics, Federal University of Technology of Paraná State (UTFPR), Brazil. He received his PhD in Engineering (Information Systems) from Federal University of Santa Catarina in 1996 and his MSc and BSc in Electrical Engineering from UTFPR in 1990 and 1984, respectively. He is the Founder and Head of Bioinformatics Laboratory. His current research interests are bioinformatics, evolutionary computation and reconfigurable computing and parallel computing.

1 Introduction

Proteins are organic components of all living beings and composed by smaller structures known as amino acids, bonded on a chain-like structure. The linear sequence of amino acids (known as primary structure of a protein) has all the information for generating a unique tri-dimensional structure (tertiary structure). Once correctly folded the amino acids chain, the protein will be ready to have a biological function. A wrongly folded protein not only can loose its biological function, but also be very harmful to the organism, giving origin to several diseases. Scientists have tried to unveil and simulate the exact way a protein folds after during its synthesis in the ribosome of the cell. Despite the huge efforts in this area, the way proteins fold into their functional structure is still an open issue.

Therefore, predicting protein structure has become a central problem in bioinformatics, once simulating protein folding on a realistic way is too complex and expensive even with the modern computational resources.

To overcome such problem, since long ago, researchers (mainly in the bioinformatics area) have proposed alternative models to simplify the structure of the proteins. Several approaches were proposed, ranging from lattice-based to free-energy models. For a comprehensive overview of computational models of protein structure, see Lopes (2008).

The simplest representation for a protein structure is the model proposed by Dill (1985) and known as hydrophobic/polar (HP). In this model, the amino acids present in a protein are converted to monomers H (hydrophobic) or P (hydrophilic or polar), depending on its

affinity to water. Each monomer of the chain is placed on a quadratic or cubic lattice and their movements are constrained by the lattice. Despite the simplicity of the HP model, it was shown that using it for protein structure prediction takes to a HP-hard problem (Ngo et al., 1994). That is, there is no polynomial time algorithm to solve it. Therefore, many heuristic approaches have been proposed, such as: neural networks (Stillinger and Head-Gordon, 1995), multicanonical Monte Carlo (Irbäck et al., 1997), simulated tempering (Irbäck and Potthast, 1995), PERM (Hsu et al., 2003), conventional metropolis type Monte Carlo procedures (Stillinger and Head-Gordon, 1995) and some evolutionary computation-based techniques (Scapin and Lopes, 2007; Lopes and Bitello, 2007).

The model studied in this paper is based on the HP model, but the position of the monomers is not restricted to the crossings of the lattice. Instead, they can be positioned anywhere in a plane, connected by bounds. This model is known as toy model or AB model and was proposed by Stillinger et al. (1993). In the same way as in the HP model, the AB model also considers only two types of amino acids: A (hydrophobic) and B (polar).

Both the HP and AB models were proposed to solve the protein structure optimisation problem, with different levels of abstraction. All models proposed in the literature have a common goal: they want to find the best folding for a protein with the lowest energy state.

Differential evolution (DE) (Price et al., 2005; Chakraborty, 2008) is an optimisation method, pertaining to the area of evolutionary computation. DE has been successfully used for difficult optimisation tasks, including protein structure prediction (Lopes and Bitello, 2007).

The objective of this work is to use DE algorithm as an alternative to solve the protein structure optimisation problem based on the AB model. Two approaches were implemented, a regular sequential implementation and a parallel implementation.

In the next sections, we will describe the AB model and both implementations, comparing their results with other algorithms published in the literature.

2 Differential evolution

DE (Price et al., 2005; Storn and Price, 1997) is a heuristic optimisation method from the evolutionary computation field and was proposed for solving polynomial fitting problems. The basic idea of DE is the use of difference vectors for generating perturbations in a population of vectors. DE is conceptually simple, easy to implement and has proven to be flexible and achieve good solutions for many interesting problems (Plagianakos et al., 2008).

The application of DE to real-world problems requests the definition of the following control parameters:

- Population size (*pop*): represents the number of candidate solutions that the algorithm will handle at the same time.
- Dimension of solutions (n_{Dim}): defines the length of the vectors that represent individuals. Each element of the vectors 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): applied to the vector resulting from the difference between pairs of vectors (say, X_2 and X_3). Typically F is a real-valued parameter in the range [0, ..., 2].
- Crossover probability (CR): 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. This latter vector (X_i) can be either randomly chosen or the one with the best fitness found up to the moment (X_{best}). The final result of the operation is a candidate vector ($X_{candidate}$).
- Strategy for vector operations: several different evolution strategies (vector operations) were proposed (Price et al., 2005), 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 iterations.

After defining these control parameters, the initial population is randomly created. In order to have the coverage of the search space as even as possible, a random number generator with long period and uniform probability distribution should be used.

Next, as in all evolutionary computation algorithms, the fitness function of each individual of the population have to be evaluated, according to the specific meaning of the elements of the vector. While a stop criterion, previously set, was not met, the following loop is repeated:

- For each individual X_i of the population do the following. Choose at random three other vectors of the population, for instance, X_1, X_2 and X_3. Vector X_1 could be the one with the best fitness value up to now (X_{best}).
- For each element of vector X_i, generate a random number *rnd* in the range [0, ..., 1]. If $rnd \leq CR$, the current element of the vector substitutes the corresponding element of the same index in vector $X_{modified}$ (described in the next item) generating, at the end, a new individual, $X_{candidate}$. This operation is somewhat equivalent to the crossover operator, commonly used in genetic algorithms. There are several variations of this procedure.
- For each element of vector X_1, apply the selected strategy for vector operation. The result of this operation is $X_{modified} = F(X_2 - X_3)$. The elements of the vector are always constrained by upper and lower bonds. The vector operations over X_1 are equivalent to the mutation operator of genetic algorithms.

- Evaluate the fitness of vector $X_{candidate}$ according to its meaning to the problem in hand.
- Considering here a minimisation problem, if this fitness is smaller than the fitness of X_i, that is, $f(X_{candidate}) < f(X_i)$, vector X_i is substituted by $X_{candidate}$. This operation is equivalent to the selection procedure of genetic algorithms.
- If $X_{candidate}$, just included in the population, has fitness smaller than X_{best}, then the new X_{best} will be $X_{candidate}$.

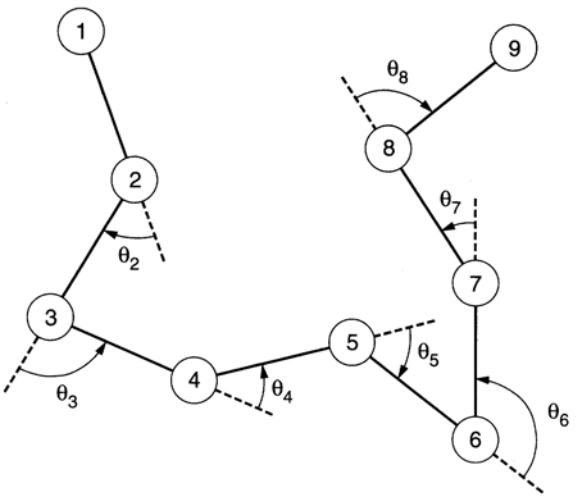
3 The AB off-lattice model

This model for representing the structure of a protein was first proposed by Stillinger and and Head-Gordon (1995). A protein is formed by a combination of 20 possible standard amino acids, although this model considers only two species of monomers, 'A' which represents hydrophobic amino acids and 'B' which represents polar amino acids. Using {A, B} as alphabet, the primary structure of any complex protein is represented by a sequence of A's and B's. The monomers are connected by unit-length (distance = 1) bonds each other and spatially grouped in such a way that the bonds between them form an angle, relative to its predecessor. For any protein structure composed by n-monomers represented with the AB model, n-2 bend angles will be needed. These angles are defined in range $-\pi \leq \theta_i \leq \pi$.

Figure 1 shows the representation of a hypothetical protein composed by nine amino acids, each one bonded to the next of the chain.

It is worth to recall that the AB model is a very simplified abstraction of a real protein structure, although it can be useful for verifying some of the properties of real-world proteins.

Figure 1 Generic representation of a hypothetic 9-mer protein structure with its bended angles



The model defines energy values for monomers: '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 of monomers give rise to different potential energy values (C): for AA bonds the energy is 1, meaning that AA monomers tend to attract strongly each other; BB bonds have energy +1/2, meaning that they have tendency of attracting each other weakly and AB or BA bonds have energy –1/2, meaning that when they are bonded they have tendency for weak repulsion.

The energy for a protein structure with n monomers (n-mers) is given by equation (1):

$$\phi = \sum_{i=2}^{n-1} V_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} V_2(d_{ij}, \xi_i, \xi_j) \tag{1}$$

The above expression postulates two types of intermolecular potential energies. The first term (V_1) depends only on the angle between monomers and represents the backbone potentials; the second term (V_2) represents the potential energy present in the non-bonded interactions and it is known as the Lennard-Jones potential. These terms are defined by equations (2) and (3), respectively:

$$V_1(\theta_i) = \frac{1}{4.(1-\cos\theta_i)} \tag{2}$$

$$V_2(d_{ij}, \xi_i, \xi_j) = 4.\left(d_{ij}^{-12} - C(\xi_i, \xi_j).d_{ij}^{-6}\right) \tag{3}$$

where

$$C(\xi_i, \xi_j) = \frac{1}{8.(1+\xi_i+\xi_j+5.\xi_i.\xi_j)}$$

is the potential energy due to the interaction between monomers i and j and d_{ij} is the distance between these monomers in the chain, such that $i < j$.

4 Methodology

For applying DE to the protein structure optimisation problem, the individuals are encoded directly with the n-2 angles between the monomers. Angles are defined based on the axis between the previous monomer and the current on (see Figure 1), thus limiting values in the range $[-\pi, ..., \pi]$, meaning that 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 interval.

As mentioned in Section 2, there are several evolution strategies for dealing with the vectors. That is, how individuals of the population, represented by vectors, are perturbed.

The weighing factor (F) is used to calculate the weighted difference between two or more vectors, this parameter strongly influences the result of the vector operation, increasing or decreasing the convergence of the population. Basically, there are two ways of combining new vectors. DE will always get one vector from the population and then randomly choose the second or third vector,

depending on how many vectors the strategy uses and combine with the first one applying F and CR. Another strategy can combine the selected vector with the best vector so far on the population. Empirically, Storn and Price (1997) discovered that randomly chosen vectors, instead of the best vector, would increase the diversity of the population along iterations. On the other hand, using the best vector would lead to fast convergence.

The crossover type can be binomial (*Bin*), when every element of the vector has the same probability for crossover (that is, CR) or exponential (*Exp*) when crossover is done while a randomly chosen value is less or equal to CR.

According to the above mentioned parameters, the evolution strategies for DE are parametrically defined as a triplet: selection type/vectors selected/crossover strategy. The selection type can be either *Rand* or *Best*, the number of vectors selected is usually 1 or 2 and the crossover strategy can be *Bin* or *Exp*.

The evolution strategies that are typically used in DE problems are: *Rand/1/Exp*, *RandtoBest/1/Exp*, *Best/1/Exp*, *Best/2/Exp*, *Rand/2/Exp*, *Best/1/Bin*, *Rand/1/Bin* and *RandtoBest/1/Bin*.

4.1 Benchmarks and parameters tuning

The first approach taken to solve the protein structure optimisation problem was find DE strategies, F and CR values capable of leading DE to find the best results. We used some of the strategies and parameters suggested by Storm and Price (1997) and Feoktistov (2006) in their respective works. Both works suggest that DE is not too sensitive to CR than to F, therefore, during the evolution we decided to set $CR = 0.85$. According to the references mentioned, the following set of values for F was tested: {0.4, 0.6, 0.8, 0.95}. Also, three evolution strategies were tested in this work: *Rand*/1/*Exp*, *RandtoBest*/1/*Exp*, *Best*/1/*Exp*.

We used a set of benchmark sequences of monomers that was previously used in other works in the literature. These sequences were used in all experiments in this paper. The sequences of 'As' and 'Bs' are described in Table 1, 'N' is the number of monomers and '$E_{\min}$' is the minimum energy known to date (see Section 6 for details).

Table 1 Benchmark of sequences

Sequence	*N*	E_{min}
ABBABBABABBAB	13	–3.2939
BABABBABABBABBABABBAB	21	–6.1976
ABBABBABABBABBABABBAB ABBABBABABBAB	34	–10.7001
BABABBABABBABBABABBAB ABBABBABABBABBABABBAB ABBABBABABBAB	55	–18.5154

The sequential version of DE (see Section 4) was run ten times for each set of parameters. The best values were stored and the average was taken. The results for each sequence are shown in Figures 2 to 5. In these plots, the vertical axis is the average of the best values found for the energy of the folding and the horizontal axis is the tested values of F.

From the plots in the figures below, it is possible to observe that for all, but the longest sequence, the strategy that performed best most times was *Rand/1/Exp*. For the sequence of 55 monomers, there is no clear definition of which is the best strategy. Regarding the value of F, a clear tendency was observed: the larger its value, the better the results. Again, for the longest sequence, there was no consistency in this observation.

Overall, these preliminary experiments for tuning DE showed how difficult is it to find suitable values for the running parameters. As a matter of fact, optimal values is problem-dependent (Brest et al., 2008), not only for DE, but also for most evolutionary computation algorithms.

Figure 2 Average best results for the 13 monomers sequence (see online version for colours)

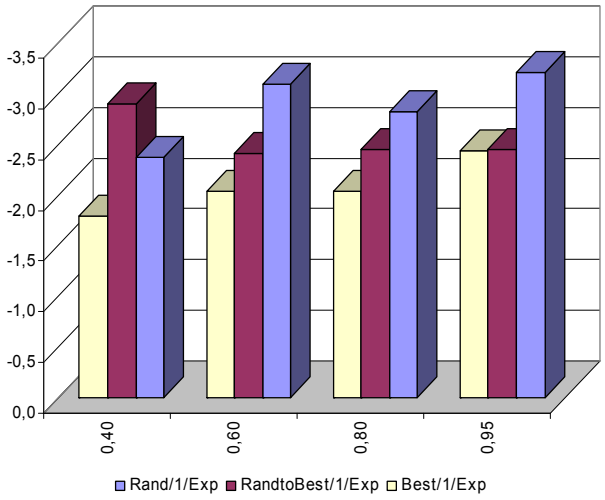


Figure 3 Average best results for the 21 monomers sequence (see online version for colours)

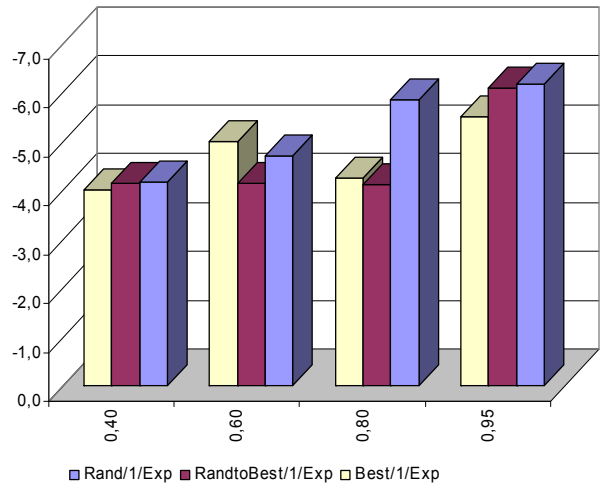


Figure 4 Average best results for the 34 monomers sequence (see online version for colours)

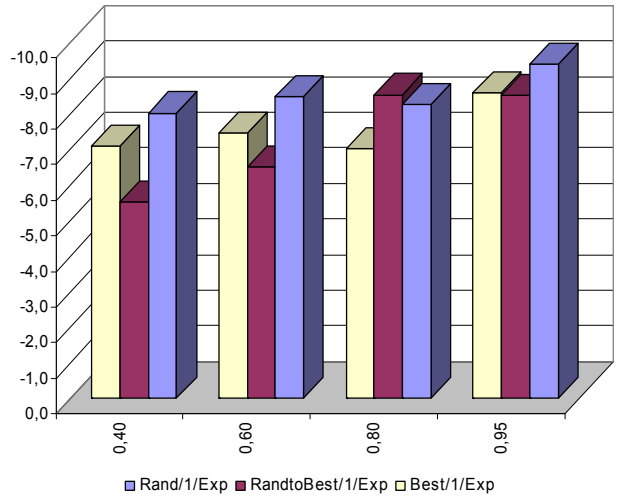
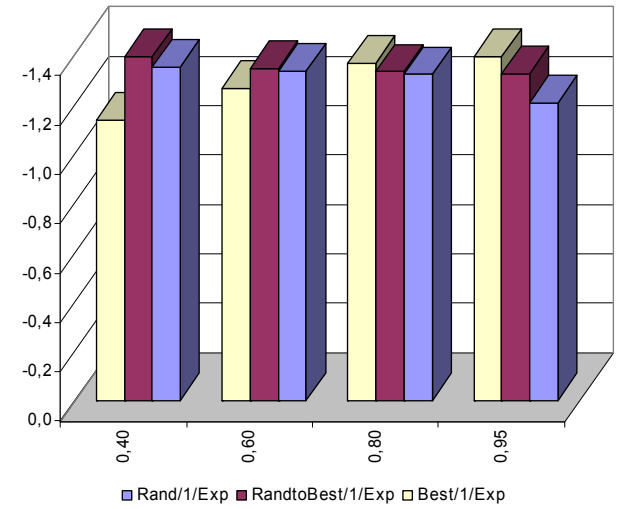


Figure 5 Average best results for the 55 monomers sequence (see online version for colours)



5 Sequential implementation

The first implementation of the DE algorithm to solve the sequence optimisation problem was done based on the algorithm proposed by Storm and Price (1997). The algorithm was initialised with a population of five individuals, each one n-2 dimensions, corresponding to the folding angles of n-mers protein.

Each population evolved for 1,000,000 iterations, the weight factor (F) equals and crossover factor (CR) were tested using different values (see next section).

We empirically observed that using a fixed weight factor (F) turned out not to be the best approach, because the algorithm got stuck in local minima, unable to evolve good solutions. Therefore, a tuning approach for F was devised, based on the number of iterations the algorithm did not improve the best solution (considering a number of decimal digits). We set a cut point in the 5th decimal digit and 500 iterations without improvement to toggle the weight factor F to a random number between 0.5 and 3.95. Later, when the algorithm is capable of improving solutions again, F is toggled back to the original value.

We also implemented another strategy that helps to escape from local minima, called population explosion. This strategy was based on similar approach that was proved to be useful for particle swarm optimisation (Hembecker et al., 2007). When the population stagnated (even after toggling F) for 10,000 iterations, it is 'exploded'. That is, the population re-started from scratch, but preserving the best-to-date individual before explosion.

6 Parallel implementations

The parallel DE algorithm was implemented aiming at improving its performance. The parallel implementation was based on the MPICH-2 (http://www.mcs.anl.gov/research/projects/mpich2/), a portable and updated implementation of the message passing interface (MPI) (Gropp et al., 1999). MPI is a communication protocol widely used for parallel implementations of scientific applications. Basically, MPI provide standardised means of communication and control between processes running in (the same or) different machines. By parallelising the sequential DE implementation using MPI standards we divided the computational load and improved the overall performance.

Two different topologies were proposed for the parallel DE: master-slave (MS) and ring-island (RI). The first approach is to distribute between processes the computation of fitness, because this is the part with highest computational cost of the algorithm. The second implementation will try to improve the results, using an island structure composed by some process that will divide the workload for the fitness calculation but they also communicate with another island sharing its best individual. Both implementations will be detailed below.

6.1 MS approach

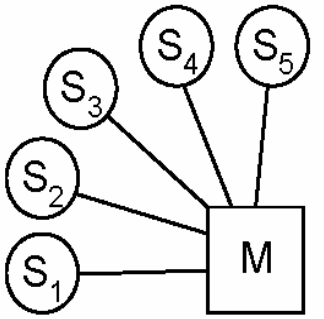
The MS approach requires six MPI processes in order to distribute the computation of the fitness. Figure 6 shows a diagram of the MS approach: the square represents the master process; the circles are the slaves and lines between them are the communication path.

In our cluster, each process runs in a physical processor (either in the same machine or not). One process is called master and the remaining five are the slaves. The master process is responsible for controlling and distributing the tasks to the slaves, as well as gathering results and executing operations over vectors. This process contains the information about the current strategy used by the DE algorithm and the set of predefined parameters for the run (F and CR).

When the algorithm is started the master process creates the initial population and starts the slaves. Once started the slaves get idle waiting for tasks do; the master distributes evenly to the slaves the population of individuals of which

the fitness (energy of the folding) have to be computed. Each slave can receive a single individual or a chunk of individuals, determined during the program initialisation. Once the fitness of the individual (or group of individuals) is computed, slaves return the result to the master. This process is synchronous; since the master waits for the reply of all slaves before continue the DE algorithm.

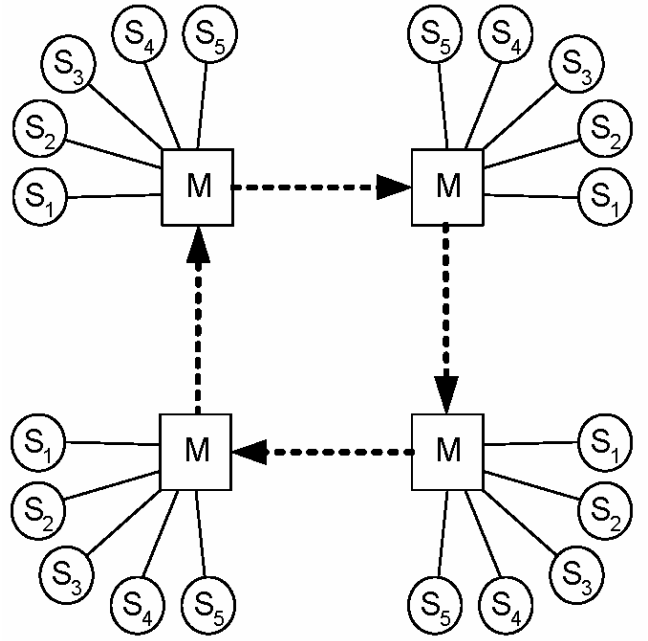
Figure 6 MS approach for the parallel DE



6.2 RI approach

The RI approach of parallel DE was done using four different islands, connected with each other using a ring model, as shown on Figure 7. In this figure, dotted lines represent the flow of migration between islands.

Figure 7 RI approach for the parallel DE



Each island has the same structure as the MS approach, running six processes. However, each island has its own set of parameters and evolution strategies. Islands are totally independent from each other, that is, they can run its own set of parameters and evaluate its individuals even if the other islands are down. A migration policy was defined: from time to time the best individual of a given island can migrate to the next island and replaces randomly one of the individuals on that specific population. This migration is done following the precedence established by the ring structure shown in Figure 7.

In our implementation, we created two possible migration policies. The first is called forced migration: from time to time individuals of an island will migrate to the next island and so forth, this migration done every time the master reaches 7,500 iterations. The second migration policy takes place when stagnation is detected in DE, according to the same strategy used for the sequential approach (see Section 4).

This approach is asynchronous, because each island can process a different number of individuals and have different parameters. However, migration only occurs only when the island has finished its current generation.

7 Results

The best results using both implementations, sequential and parallel are shown in Table 2. In this table, there are also results obtained by other authors using different methods: first (E_{PERM}) is a pruned-enriched Rosenbluth method – PERM, by Hsu et al. (2003), next (E_{min}) is the minimum energy obtained by the same method with subsequent conjugate gradient minimisation. E_{ground} is the putative ground state energy obtained by Stillinger and Head-Gordon (1995) using a Monte Carlo method hybridised with Newtonian conjugate gradient minimisation. The second group corresponds to results obtained in this work: DESeq is the sequential DE implementation, DE-MS and DE-RI refer to the two approaches of the parallel DE implementations, MS and RI, respectively.

Table 2 Best results obtained and comparison with other works

N	E_{PERM}	E_{min}	E_{ground}
13	–3.2167	–3.2939	–3.2235
21	–5.7501	–6.1976	–5.2881
34	–9.2195	–10.7001	–8.9749
55	–14.9050	–18.5154	–14.4089
N	*DESeq*	*DE-MS*	*DE-RI*
13	–3.1999	–3.1999	–3.2924
21	–6.19799	–6.19799	–6.19799
34	–9.29173	–9.15178	–9.68382
55	–11.52403	–13.7471	–14.68478

In order to evaluate visually the quality of the foldings produced by the DE algorithm in this work, the best results, previously mentioned, were used to draw the planar form of the sequence (conformation). A program in MATLAB was developed to convert the string of angles into $\{x, y\}$ coordinates and plot the structure. The larger dot represents the start of the sequence, which can be either 'A' or 'B' monomers, black dots represent 'A' monomers and the yellow dots represent 'B' monomers. Recall that the energy of the folding is a function of the proximity of monomers, especially the 'A' monomers. Therefore, compact structures

tend to have lower energy levels than those structures more dispersed. Figures 8 to 19 show the best foldings obtained with the three DE implementations (sequential – DESeq, parallel MS – DE-MS and parallel RI – DE-RI), for sequences with 13, 21, 34 and 55 monomers.

Figure 8 Best folding for $N = 13$ (DESeq) (see online version for colours)

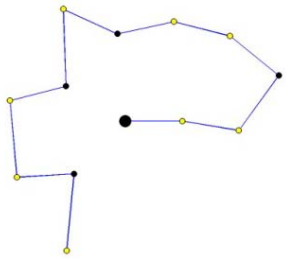


Figure 9 Best folding for $N = 13$ (DE-MS) (see online version for colours)

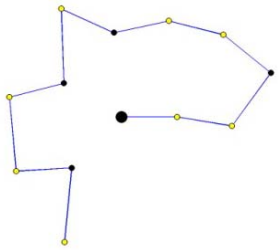


Figure 10 Best folding for $N = 13$ (DE-RI) (see online version for colours)

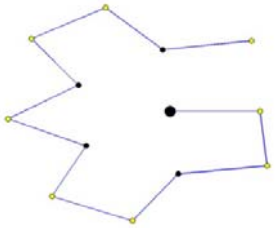


Figure 11 Best folding for $N = 21$ (DESeq) (see online version for colours)

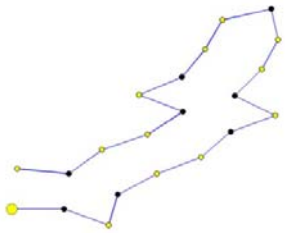


Figure 12 Best folding for $N = 21$ (DE-MS) (see online version for colours)

Figure 13 Best folding for $N = 21$ (DE-RI) (see online version for colours)

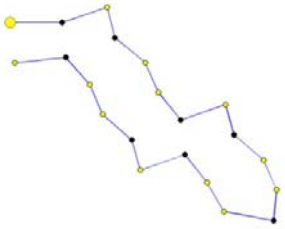


Figure 14 Best folding for $N = 34$ (DESeq) (see online version for colours)

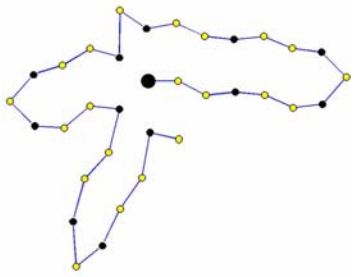


Figure 15 Best folding for $N = 34$ (DE-MS) (see online version for colours)

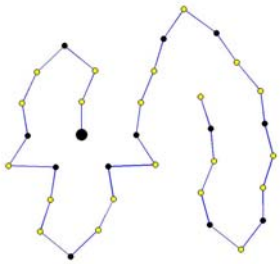


Figure 16 Best folding for $N = 34$ (DE-RI) (see online version for colours)

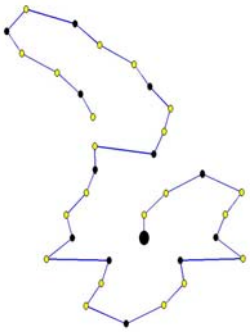


Figure 17 Best Fitness $N = 55$ (DESeq) (see online version for colours)

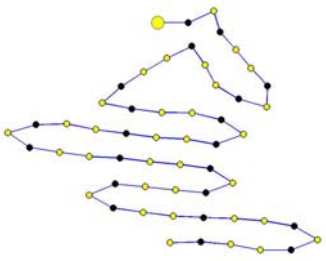


Figure 18 Best folding for $N = 55$ (DE-MS) (see online version for colours)

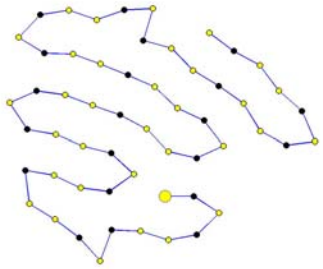
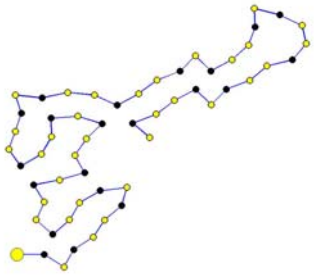


Figure 19 Best folding for $N = 55$ (DE-RI) (see online version for colours)



8 Analysis of results

The analysis of results shown at the previous sections (Table 2 and Figures 8 to 19) indicates an evolution of quality, from the sequential implementation to the parallel RI model.

During the sequential tests for $N = 13$ the best energy value obtained was –3.1999. This result turned out to be a strong local minimum that could only be broken by the parallel DE-RI implementation. Possibly, this was due to the different evolution strategies, different parameters and the migration between the islands.

For the $N = 21$ sequence the results of the sequential implementation and the other two parallel implementations were basically the same. Despite being longer than the previous sequence, it seems to be easier to fold than it. For this particular sequence DE seems to be more efficient than the other methods, since it obtained a slightly better result.

For the $N = 34$ sequence, the results obtained by our implementation were only 13%, 14% and 9% above the minimum energy known. Notice that both DESeq and DE-RI still obtained better results than the PERM implementation.

For the hardest problem, $N = 55$ monomers, DE was not able to find good results when compared with the minimum energy known. However, our results are, in general, comparable to results of the PERM method as well as of the ground energy found by Stillinger and Head-Gordon (1995). The DE-RI implementation obtained quite similar results to these previously mentioned results.

9 Conclusions

Several protein folding problems were tested using DE algorithm, in sequential and parallel versions. DE turned out to be an interesting method to predict protein structure using the AB model. The parallel DE versions had superior performance when compared with the sequential one. Particularly, the RI approach was more efficient than the MS approach.

As previously described for smaller proteins DE was able to find the ground state energy values. However, as the number of amino acids of the sequence increases, the complexity of the problem increases exponentially and DE looses performance. Anyhow, DE (the parallel versions) got close to the best values obtained for the $N = 34$ protein, finding energies lower than some previous implementations, although it was not able to find the ground states for $N = 34$ and $N = 55$.

It is worth to recall that the current optimal results were obtained with specialised methods hybridised with local search procedures and DE is a general-purpose method. Also, no special operators or problem-dependent approaches were used to improve performance of DE.

The overall comparison of results suggests that DE may be a promising method for solving the structure optimisation problem for small protein sequences; although for long sequences the method degrades performance. Possibly, this is due to conjunction of several factors: the huge search space that grows exponentially as the number of amino acids increase; the nature of the problem which is highly constrained and the premature convergence of the algorithm to local minima and difficulty in escaping from there.

These facts suggest the need for special manipulations of the vectors, which could take into account not only the physical constraints of the foldings, but also some biological knowledge. In the same way, hybridisation seems to be needed, especially towards the use of concomitant local search procedures.

The promising results reported here encourages further research, mainly towards the use of special operators, so as to improve results for long protein sequences.

Acknowledgements

This work was partially supported by the Brazilian National Research Council – CNPq, under Research Grant No. 309262/2007-0 to H.S. Lopes.

References

Brest, J., Zamuda, A., Bošković, B, Greiner, S. and Žumer, V. (2008) 'An analysis of the control parameters' adaptation in DE', in Chakraborty, U.K. (Ed.): *Advances in Differential Evolution*, Springer, pp.89–110.

Chakraborty, U.K. (2008) *Advances in Differential Evolution*, Springer.

Dill, K.A. (1985) 'Theory for the folding and stability of globular proteins', *Biochemistry*, Vol. 24, pp.1501–1509.

Feoktistov, V. (2006) *Differential Evolution: In Search of Solutions*, Springer.

Gropp, W., Lusk, E. and Thakur, R. (1999) *Using MPI-2: Advanced Features of the Message-Passing Interface*, MIT Press.

Hembecker, F., Lopes, H.S. and Godoy, W. Jr. (2007) 'Particle swarm optimization for the multidimensional knapsack problem', *Proceedings of Adaptive and Natural Computing Algorithms (ICANNGA)*, Warsaw, Poland, Lecture Notes in Computer Science, Springer, Part I, Vol. 4331, pp.358–365.

Hsu, H.P., Mehra, V. and Grassberger, P. (2003) 'Structure optimization in an off-lattice protein model', *Physical Review E*, Vol. 68, No. 3, pp.037703–037707.

Irbäck, A. and Potthast, F. (1995) 'Studies of an off-lattice model for protein folding: sequence dependence and improved sampling at finite temperature', *Journal of Chemical Physics*, Vol. 103, pp.10298–10305.

Irbäck, I., Peterson, C., Potthast, F. and Sommelius, O. (1997) **'**Local interactions and protein folding: a 3d off-lattice approach**',** *Journal of Chemical Physics*, Vol. 107, pp.273–282.

Lopes, H.S. (2008) 'Evolutionary algorithms for the protein folding problem: a review and current trends', in Smolinski, T.G., Milanova, M.M. and Hassanien, A-E. (Eds.): *Applications of Computational Intelligence in Bioinformatics and Biomedicine: Current Trends and Open Problems*, Springer-Verlag, Vol. 1, pp.297–315.

Lopes, H.S. and Bitello, R. (2007) 'A differential evolution approach for protein folding using a lattice model', *Journal of Computer Science and Technology*, Vol. 22, No. 6, pp.904–908.

Ngo, J.T., Marks, J. and Karplus, M. (1994) 'Computational complexity, protein structure prediction and the Levinthal paradox', in Merz, K. Jr. and LeGrand, S. (Eds): *The Protein Folding Problem and Tertiary Structure Prediction*, Birkhauser, pp.433–506.

Plagianakos, V.P., Tasoulis, D.K. and Vrahatis, M.N. (2008) 'A review of major application areas of differential evolution', in Chakraborty, U.K. (Ed.): *Advances in Differential Evolution*, Springer, pp.197–238.

Price, K.V., Storn, R.M. and Lampinen, J.A. (2005) *Differential Evolution: A Practical Approach to Global Optimization*, Springer.

Scapin, M.P. and Lopes, H.S. (2007) 'A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model', in Ang Yang, Yin Shan and Lam Thu Bui (Eds.): *Success in Evolutionary Computation*, Springer-Verlag, pp.205–224.

Stillinger, F.H. and Head-Gordon, T. (1995) 'Collective aspects of protein folding illustrated by a toy model', *Physical Review E*, Vol. 52, pp.2872–2877.

Stillinger, F.H., Head-Gordon, T. and Hirshfeld, C.L. (1993) 'Toy model for protein folding', *Physical Review E*, Vol. 48, pp.1469–1477.

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