A Hybrid Genetic Algorithm for the Protein Folding Problem Using the 2D-HP Lattice Model

Heitor S. Lopes and Marcos P. Scapin

Bioinformatics Laboratory, Federal University of Technology – Paraná, Av. 7 de setembro, 3165 80230-901 Curitiba, Brazil hslopes@utfpr.edu.br

1 Introduction

Proteins perform vital functions in all living beings. The function of a protein is determined after being folded into a specific 3D structure under certain physiological conditions. This structure is called its native conformation. Therefore, understanding how proteins fold is of great importance for Biochemistry and Biology. Proteins are formed by amino acid chains that can be roughly classified as hydrophobic (aversion to water) or hydrophilic (affinity to water).

It is commonly accepted that the main driving force of the formation of the structure of a protein is the internal interaction of hydrophobic residues. This means that the amino acids with hydrophobic side-chains in a protein sequence tend to be grouped together to form a hydrophobic core. The hydrophilic amino acids tend to be pushed away from the core and interact with surrounding solvent molecules.

According to Anfinsen's experiments performed in the 1960s, the 3D structure of a protein can be predicted by the analysis of its amino acid sequence. The task of predicting a protein tertiary structure is called the Protein Structure Prediction Problem (PSP), and the way it folds into its native conformation is known as Protein Folding Problem (PFP). Both are current problems in the modern Computational Biology that has drawn attention of Computer Science experts.

The exhaustive search of the conformational space of a protein is not possible with the current computational technology, even for small proteins. Therefore, simple lattice models have been proposed to decrease the complexity of the problem. However, even so, the PFP is NP-complete and, therefore, intractable for most real-world instances [1]. Consequently, heuristic optimization methods seem to be the most reasonable algorithmic choice to solve this problem. Among the many approaches to this problem, evolutionary computation methods have played an important role, since they are reputed as efficient

H.S. Lopes and M.P. Scapin: *A Hybrid Genetic Algorithm for the Protein Folding Problem Using the 2D-HP Lattice Model*, Studies in Computational Intelligence (SCI) **92**, 115–134 (2008)
www.springerlink.com

search and optimization methods. Genetic Algorithms (GAs) do not guarantee that the global optimum will be found, but, during the evolution, it is powerful enough to find quasi-optimal solutions [2]. Therefore, many researchers have already applied GAs to the PFP [3–7]. GAs have achieved the most promising results in this area, although other methods, such as differential evolution [8] and ant colony optimization [9] have also been used.

This chapter is structured as follows: first we present a brief definition of the 2D hydrophobic-polar (2D-HP) model, followed by a detailed description of the hybrid genetic algorithm for the protein structure prediction problem using the 2D-HP model. Also, we describe the improved features of the algorithm designed to improve its predictability. Next, a complete parameter analysis is done aiming to understand their influence on the performance of the GA. We evaluated the algorithm with a benchmark of five synthetic amino acid chains, resulting in a better performance when compared with results in the recent literature. Exploring further, we also present the results of the application of the algorithm to five real-world proteins. Finally, results are discussed thoroughly and conclusions are done.

2 The 2D-HP Model

The Hydrophobic-Polar (HP) model was introduced by [10] and it is the most known and studied discrete model for protein tertiary structure prediction. This model is based on the concept that the major contribution to the free energy of the native conformation is due to interactions among hydrophobic residues. Such residues tend to form a core in the protein structure while being surrounded by the hydrophilic residues in such a way that the core is less susceptible to the environmental influence.

The HP model classifies the 20 standard amino acids in just two types: hydrophobic (H for non-polar) or hydrophilic (P for polar). This decision is taken for its simplicity, although some amino acids cannot be clearly classified as being of one of the two types [11]. Therefore, a protein is a string of characters defined over a binary alphabet $\{H, P\}$.

As it is a lattice model, the chain is embedded in a 2D square lattice and the allowed movements of the chain are restricted by the lattice. At each point, the chain can turn 90° left or right, or continue ahead. In a legal (valid) conformation, the adjacent residues in the sequence must be adjacent in the lattice and each lattice point can be occupied by only one residue.

The free energy of a conformation is inversely proportional to the number of hydrophobic non-local bonds (or H–H contact) where a H–H contact takes place if two hydrophobic residues occupy adjacent grid points in the lattice but are not consecutive in the sequence. Each such interaction contributes with -1 to the energy value. This yields two basic characteristics of real proteins: the protein fold must be compact and the hydrophobic residues are buried inside to form low-energy conformations [12]. The problem may be also

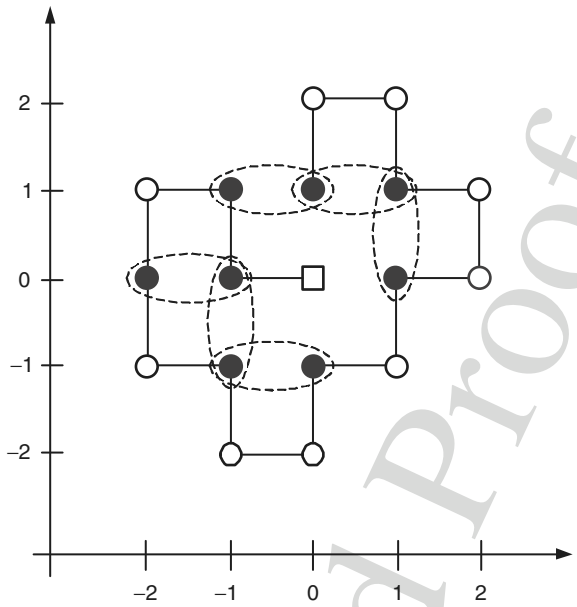

Fig. 1. Example of an amino acids chain folded in a square lattice

considered as the maximization of the hydrophobic non-local bonds, since this is the same as the minimization of the free energy of a conformation in this model. Figure 1 shows an example of an 18 amino acids-long chain embedded in a square lattice. Black and white dots represent the hydrophobic and polar amino acids, respectively, and the square dot is the first element of the chain. The chain is folded in such a way that six hydrophobic non-local bonds are formed (shown as dotted lines).

In addition to its simplicity, the folding process with the model has behavioral similarities with the real process of folding [13]. Notwithstanding, from the computational point of view, the problem of finding the native structure in using the 2D-HP model is proved to be NP-complete [1, 14].

3 Methodology

Genetic algorithms (GAs) comprise a class of heuristic methods inspired in the natural evolution of living beings and in the Darwinian principle of the survival of the fittest. GAs have been used in many computational applications including machine learning, combinatorial optimization, data mining and others. This approach is particularly interesting when dealing with NP-complete problems, whose search space is intractable by enumerative methods, or/and problems whose evaluation of the objective function is too complex, ill-defined or strongly constrained [15]. Besides the usual features of a GA, we proposed

several enhancement strategies in an attempt to improve its performance for the problem.

3.1 Chromosome Encoding

It is well-known that the dynamics and the effectiveness of a GA are strongly influenced by the way individuals are represented. Individuals represent possible solutions for a problem and, in GAs, such solutions are represented indirectly. Individuals in a GA encode a genotypical representation, while the real-world solution is the phenotypical representation, decoded from the genotype. The three most usual ways to represent a protein structure using the HP model are [16]:

(a) Cartesian coordinates: This representation is straightforward but it is too sensible to translation and rotation, in such a way that identical structures can have completely different coordinates.
(b) Internal coordinates: In this representation an amino acid position is related to the previous one, and so there are two possibilities: absolute or relative coordinates. Using absolute coordinates, a movement is defined according to the grid axes, while using relative coordinates a movement is defined according to the previous one.
(c) Distance geometry: This representation describes a structure using a matrix of all distances between every pair of points.

Most GA-based approaches so far use internal coordinates because the connectivity of the chain is implicit in the encoding, i.e., every consecutive amino acid in the chain must be adjacent in the grid. Besides, a study presented by [17] showed empirically that the use of relative coordinates in the chromosome encoding achieved better results in comparison with absolute coordinates. Therefore, in our implementation we also used relative coordinates. In a 2D space, there are only three possible moves: (R)ight, (L)eft and (F)orward. Thus, an individual encoding is defined over the alphabet $\{R, L, F\}$. Considering a N-residue long chain, the individual will have $N-1$ genes, representing the moves required to form a specific conformation.

When applying GA to solve constrained problems, depending on the representation used, invalid individuals may appear as result of the application of operators or in the initial random population. In principle, this problem can be handled in three different ways: eliminating invalid individuals, fixing them or allowing them to survive. The first approach is simple and straightforward, but possible useful genetic material can be lost and not recovered later. The second approach is interesting but usually it is computationally intensive. In the last approach, invalid individuals are allowed to survive in the population but their fitness value is decreased proportionally to the constraints violation. This penalty strategy is useful for preserving potentially interesting genetic material for further generations.

Despite the advantages of the representation chosen, it allows two or more amino acids to occupy the same position in the lattice. This fact, known as collision, leads to an illegal conformation. Therefore, the penalty strategy was used to penalize individuals that results in such unfeasible folds. These individuals can appear during evolution as result of the application of genetic operators and a penalty is added to the fitness function for every lattice point at which there is more than one amino acid. Another important reason to allow individuals that decode to unreal folds during evolution is that the shortest path from one compact legal conformation to another legal conformation may be very short if illegal conformations are allowed, when compared to the shortest path if only legal ones exist [17].

3.2 Initial Population

According to [4], the encoding using relative internal coordinates exhibits the problem that the initial population (randomly initialized) tends to have increasing number of collisions as the length of the protein increases. Therefore, the GA spends much time working with the illegal conformations before good results can be obtained. To circumvent this problem, we proposed a different technique to build the initial population, taken from another type of evolutionary algorithms, called Genetic Programming [18]. This technique, named ramped-half-and-half, does not ensure that individuals in the initial population will be free from collisions, but it tends to minimize them generating a greater diversity of conformations.

To implement this technique, the population is divided into two parts generated differently. The proportion of each part is a user-defined parameter as a percentage of the population size. The first part of the population is generated in a totally random manner, as usual. The second part is generated considering each individual as totally unfolded and applying several random mutations that varies between three and the number of genes in the chromosome. This is done in a way that this part contains a proportional number of individuals for each value, ranging from three to the number of genes in the chromosome. The minimum of three mutations was chosen because of the fact that conformations with 0, 1 and 2 mutations will have few hydrophobic contacts, and such individuals will bring very little contribution to the evolution.

The initial population created in this way will have a large diversity, a necessary condition to evolution. Besides, we empirically observed that the evolution process would be helped if a considerable number of individuals having few mutations (that is, they have large unfolded parts) is present.

3.3 Fitness Evaluation

To evaluate an individual, it is necessary to decode its genotypical representation (chromosome) defined over the alphabet $\{R, L, F\}$ to obtain the

corresponding Cartesian coordinates (phenotypical representation). This set of coordinates describes how the residues are disposed in the lattice, that is, it represents a 2D conformation. After, the conformation is evaluated by a fitness function that gives a numerical value representing the goodness of the solution for the folding problem. The fitness function proposed is the product of three terms, as shown in (1):

$$fitness = HnLB \times RadiusH \times RadiusP\ , \tag{1}$$

where $HnLB$ is the number of H–H non-local bonds (related to the free energy of the conformation), and $RadiusH$ and $RadiusP$ are terms computed using the radius of gyration of the hydrophobic and hydrophilic amino acids, respectively. These terms are explained in the next sections.

Hydrophobic Non-Local Bonds

It is believed that the hydrophobic non-local bonds are the main force that drives the protein folding process. Since the problem is treated as the maximization of the H–H contacts, for every hydrophobic non-local bond, the energy function is added by 1. The complete energy function is shown in (2):

$$HnLB = \left(\sum_{i<j} e_{v_i v_j} . \Delta(r_i - rj) \right) - (NC.PW)\ , \tag{2}$$

where NC is the number of collisions of the folding; PW is the penalty weight; and $\Delta(r_i - r_j) = 1$ if residues r_i and r_j form a non-local contact and $\Delta(r_i - r_j) = 0$, otherwise. Depending on the contact types between residues, the energy $e_{v_i v_j}$ will be e_{HH}, e_{HP} or e_{PP}, corresponding to H–H, H–P or P–P contacts, respectively. In our implementation, e_{HH} was set to 1 and the remaining, to 0.

Recalling that we are using a penalty strategy for illegal conformations, the penalty term ($NC.PW$) is subtracted directly from the energy function. This term comprises the number of grid points which are occupied by more than one amino acid, multiplied by the penalty weight that is set according to the chain length: the longer the chain, the higher is the penalty weight. In preliminary experiments with chains of different lengths we observed that for chains around 20 amino acids, $PW = 2$ was adequate. For chains of length around 50, $PW = 3$ lead to better results, while for chains around 80 amino acids, $PW = 4$ was more effective. Therefore, a straight line was interpolated using these points, and (3) shows how PW is adjusted as function of the number of amino acids (NA) of the chain

$$PW = (0.033 \times NA) + 1.33\ . \tag{3}$$

Radius of Gyration

An important issue to be taken into account, while attempting to predict the structure of a protein, is related to the energy hypersurface, i.e., how the free energy is distributed considering all possible conformations. In fact, it is not only the size of the energy landscape that matters, but also its shape. The energy landscape is protein-dependent, that is, for each protein, a different energy landscape exists. Therefore, the energy hypersurface has the dimension corresponding to the number of amino acids in the chain and can have many local minima.

The original HP model uses only the number of hydrophobic non-local bonds to evaluate a solution. Due to the nature of this discrete model, it creates large plateaus in the energy landscape on which local search cannot find a descent direction, and where it effectively performs a random search. This fact was also confirmed in our preliminary experiments and motivated the creation a modified fitness function. Using a discrete energy function (such as that of (2)), this landscape is much more difficult to be searched. In order to make the corresponding fitness landscape smoother, we propose the use of a new concept, the Radius of gyration (Rg) in the fitness function. The use of Rg in the fitness function can help the GA to escape from the plateaus. Furthermore, Rg increases the compactness of solutions: given two conformations with the same number of H–H contacts, Rg allows the fitness function to reward the most compact one, and thus makes the evaluation closer to reality. Radius of gyration is a physical concept from classical mechanics and is defined as the radial distance from a given axis at which the mass of a body could be concentrated without altering the rotational inertia of the body about that axis [19].

Bringing the Rg concept to the folding problem, it is a measure that indicates how compact a set of chained amino acids is: the more compact a conformation, the smaller its radius of gyration. Considering only the hydrophobic amino acids, Rg will measure how compact is the core formed by these residues. Equation (4) shows how Rg is computed for hydrophobic residues:

$$RgH = \sqrt{\frac{\sum_{i=1}^{NH}[(x_i - \bar{X})^2 + (y_i - \bar{Y})^2]}{NH}} \ , \tag{4}$$

where x_i and y_i are the Cartesian coordinates of the i-th hydrophobic amino acid, $\bar{X}$ and $\bar{Y}$ are the mean values of all hydrophobic and x_i and y_i, respectively, and NH is the number of hydrophobic amino acids in the chain. Recalling again that the problem is dealt as maximization, the final term that contributes to the objective function is given in (5):

$$RadiusH = MaxRadiusH - RgH \ , \tag{5}$$

where $MaxRadiusH$ is the calculated radius of gyration of the chain totally unfolded, supposing that this is the maximum value that can be reached.

This term of the fitness function regarding the hydrophilic radius of gyration (RgP) has the opposite purpose as $RadiusH$. That is, it fosters hydrophilic residues to lie far away from the hydrophobic core. The hydrophilic radius of gyration is computed in the same way as in (4), considering only hydrophilic residues, and without being subtracted from $MaxRadiusP$ (as in (5)). Using the RgP value, the formal definition of the term that contributes to the fitness function is shown in (6), and it will be always between 0 and 1:

$$RadiusP = \begin{cases} 1 & , \quad if \quad (RgP - RgH) \geq 0 \\ \frac{1}{1-(RgP-RgH)} & , \quad otherwise \end{cases} . \tag{6}$$

A positive difference, that is, $(RgP-RgH) > 0$, means that the hydrophobic residues are buried inside the conformation while the hydrophilic ones are outside. Such situation is desired and, in this case, $RadiusP$ has no influence in the fitness function, since the two first terms will be multiplied by 1. However, if the opposite is true, meaning that the hydrophobic residues are more spread than the hydrophilic residues, what is not desired, this conformation will be penalized by decreasing the fitness function value.

Overall, the use of radius of gyration concept in the fitness function makes the energy hypersurface smoother, allowing the GA to do a more efficient search.

3.4 Genetic Operators and Selection Method

Genetic operators are used to create new individuals in the population, by modifying existing ones. First, it is necessary a method for selecting individuals from the current population according to a criterion previously established. The selection method is not a genetic operator itself, but a procedure that must be executed before their application. In this implementation we used the tournament selection method. First, *tourneysize* individuals are randomly selected from the current population to take part of a tournament. Parameter *tourneysize* is usually taken as a percentage of the population size. Next, the best individual of this group is selected. This procedure is performed whenever genetic operators request individuals.

The use of problem-tailored genetic operators is commonly found in the literature, especially when the genetic algorithm is used for hard optimization problems. This is the case of the PFP, where one can find specific operators, biologically inspired or not (see, for instance, [20–22]). Apart from other characteristics, special genetic operators play a very important role in the PFP and, possibly, this is one of the main features that differs between implementations.

The first operator applied during the generation of a new population is the crossover operator. According to [2], this operator plays an important role for the PSP, since a piece of structure that has been useful for a given solution may be also useful for others. Two types of crossover were used in this

work: 1- and 2-point crossover and both are applied with the same probability during the evolution.

Mutation is another operator commonly used in evolutionary computation. In this work, two types of mutation are used. The first is the simple mutation where each gene is tested, according to the mutation probability, to verify whether or not the actual value of the gene will be changed. The second type, called Improved-Mutation, also changes a gene according to the same mutation probability. However, just after the application of this operator, the individual is evaluated by the fitness function. If the fitness increases, the operation is successful, otherwise, the operation is undone and the process continues with the remaining genes. In most times, this special type of mutation improves the fitness of an individual and, in the worst case, the fitness will be unchanged. During evolution, the choice between the two types of mutation is controlled by a user-defined parameter of the GA.

In preliminary experiments, we observed that using only the basic genetic operators did not lead the GA to achieve good results. Thus, new and more specialized operators were developed in order to aid the evolution process to obtain better results.

The first specialized operator called U-Fold because it tends to fold the conformation into an U shape so as to maximize the number of H–H contacts between the two faces of the sequence that are laid face to face. Firstly, this operator searches all the individual for the longest straight segment. The length of this segment must be, at least, 1/10 of the total protein length. After selecting the longest straight segment, all the possible folding points for that segment is checked in order to choose the best way to fold this segment such that the number of H–H contacts will be maximized.

Another specially designed operator is Make-Loops. It was implemented with the objective of increasing the number of contacts in an individual by grouping hydrophobic residues that were found in a straight line in the conformation. The same as in U-Fold, it searches for the longest straight segment inside the conformation but, in this case, its minimum length was set to be four residues to allow its correct application. After finding a valid segment, the operator finds the first hydrophobic residue and fixes its position in the lattice. Then, the next hydrophobic residue is searched, but it needs to be, at least, three residues far from the first. Besides, in order to make a loop, it is necessary that an even number of intermediary residues exist between the two hydrophobic ones, due to the lattice that is being used. After fixing the two positions, the operator generates a loop by positioning the intermediary residues for both sides in order to find the conformation that best contribute for the fitness of the individual. It may happen that both conformations are worse than the original individual, making the loops to be undone. This procedure is repeated until the end of the selected segment.

It is easy to realize that both operators U-Fold and Make-Loops can be useful only when there are long straight parts in the chain being folded. This happens only at the beginning of the evolution process. Therefore,

such operators work only in the initial generations of the GA, and are disabled as they became useless. The U-Fold and the Make-Loops operators are biologically inspired, since they aim at simulating the construction of stable secondary structures found in real folded proteins, such as β-sheets and α-helices [23].

A third operator used during the evolution is named Partial-Optimization, and is based on a concept taken from algorithms for combinatorial optimization problems, such as the TSP (Travelling Salesman Problem), and it is known as 2-opt. Here, we implemented a generalized version of the concept first proposed by [24]. The idea of this operator is to select randomly two non-consecutive residues of the protein chain and make their positions fixed in the lattice. Then, all possible conformations are evaluated, keeping the connectivity of the chain in the fixed points and changing the intermediate residues in the lattice. After this local search, the best conformation is kept. In the evolutionary computation area, algorithms that use some kind of local search strategy are considered as hybrid. Although there is no consensus on this issue, we prefer to consider the proposed genetic algorithm as hybrid to make clear this important feature.

The Partial-Optimization operator must be used with parsimony since, depending on the length of the internal segment between the two selected residues, this operation may become computationally expensive. It is easy to verify that the number of possibilities increases exponentially as the number of intermediate residues increases. This operator is applied to all the population during the evolution, according to a user-defined probability. The length of the intermediary segment ($POsize$) is also an user-defined parameter of the GA.

In Fig. 2, it is presented an example of the application of the Partial-Optimization operator to the sequence PHHPPPPHHPPHPP. In the figure, black and white dots represent, respectively, hydrophobic and polar amino acids. Arrows indicate the points selected to be kept fixed. The left part of the figure shows the original conformation with no H–H contacts while in the right, after the application of the operator, the new conformation presents two H–H contacts.

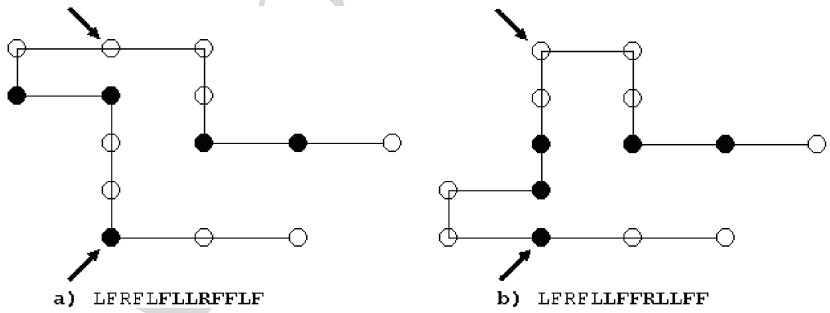

Fig. 2. Example of the application of the Partial-Optimization operator

3.5 Improvement Strategies

Preliminary tests have shown that both the proposed fitness function and the special genetic operators have helped the GA to achieve good performance. Even so, it is impossible to assure that the GA will not get trapped in a local maximum. When reaching a local maximum, depending on the running parameters, the genetic diversity of the population can be lost rapidly. This phenomenon, known as convergence, has the undesirable effect of preventing further improvement. In this situation, the only possible evolution is due to mutation, by chance. Therefore, it is useless to go on running the algorithm. Instead of stopping and restarting the algorithm from the scratch, we devised a new strategy, called Decimation-and-Hot-Boot (DHB), described as follows.

Throughout generations an elitist strategy is used jointly with the tournament selection method. This strategy always copies the best-fitted individual of a generation to the next one. If an even-better individual is not found after many generations, this is a clear signal that the population has possibly converged and, consequently, the evolution has stagnated. In this work, the number of generations with no improvement necessary to trigger DHB was fixed to 10% of the total number of generations. At this point, 50% of the population is deleted (decimated) and a number of individuals is generated according to the method mentioned in Sect. 3.2.

Applying the DHB strategy makes the population to be restarted with a large genetic diversity giving chance for the evolutionary process to continue. At the restart, it must be taken into account the fact that all the newly created individuals probably will have very low fitness values compared to those individuals that survived decimation. Obviously, it is necessary to change running parameters temporarily so as to decrease the selective pressure (towards the best individual) in such a way that all individuals can have the opportunity to evolve. This is accomplished by increasing the frequency of application of the Improved-Mutation and decreasing the tournament size. The net effect of these changes is a fast improvement in the average fitness of the population in few generations. Hopefully, this strategy can contribute to find better individuals than before. The decimation strategy is applied whenever the no-improvement counter reaches the preset value and the maximum number of generations was not reached.

While generating and evaluating the last population, the Improved-Mutation and Partial-Optimization operators are to all the individuals with 100% probability. Although expensive, this local search operation is done once, as a last chance to improve further the best solution found by GA.

4 Computational Experiments and Results

The implementation of the previously described GA features, resulted in a software system called "HGAPF"(Hybrid Genetic Algorithm for Protein Folding), which was tested using several amino acid chains to verify its

efficiency. The system was developed using Borland Delphi 7, under Microsoft Windows XP server running in a PC desktop with Athlon XP-2.4 processor and 512 Mbytes of RAM.

4.1 Parameters' Adjustment

Before running HGAPF with the benchmarks, a number of experiments was done for fine-tuning the GA parameters. For these experiments we used a real-world protein, known as 1qql, extracted from the Protein Data Bank (PDB) [25] and converted to the HP model (see Sect. 4.3 for more details). The Improved-Mutation operator is not used in the fine-tuning experiments, except when explicitly mentioned. The following standard parameters (and ranges) of the GA were tested: population size – *popsize* (200 and 500), number of generations – *gen* (100, 200 and 300), tourney size – *tourneysize* (3 and 5% of the population), probability of crossover – p_{cross} (70 and 90%), probability of mutation – p_{mut} (2, 5 and 8%). For each of these 72 experiments, 100 independent runs were done with different random seeds. Both the maximum number of H–H bonds ($maxHH$) and the average number of H–H bonds ($avgHH$) were considered. The two better combinations of parameters differ only in the number of generations, giving slightly different results. The standard parameters were then set to: $popsize = 500$, $gen = 300$, $tourneysize = 3\%$, $p_{cross} = 90\%$ and $p_{mut} = 2\%$. Using these parameters, the HGAPF took an average of 35.3 s for running.

Using the standard parameters, the Improved-Mutation operator was tested using the following probabilities of being selected among the two mutation operators ($p_{selectIM}$: 10, 30, 50, 70, 90 and 100%). This probability should not be confounded with p_{mut}, which is the probability of using a mutation operator, while $p_{selectIM}$ is the probability of selecting Improved-Mutation instead of the regular mutation operator. The results of 100 independent runs for each experiment showed that, independently of the value of $p_{selectIM}$, the use of Improved-Mutation always improve results. However, increased values for $p_{selectIM}$ lead to a faster convergence of the GA. Therefore, $p_{selectIM}$ was set to 10%.

Next, the probability of applying the U-Fold (p_{UF}) and the probability of applying the Make-Loops (p_{ML}) operators were tested, both in the range from 10% to 100%. Again, the standard parameters were used with the Improved-Mutation operator turned off. Besides $maxHH$ and $avgHH$, the average number of generations in which each operator was used was also observed. For each of the 20 experiments, again 100 independent runs were done. The best results were found when both probabilities were set to 10%, which maximized, at the same time $maxHH$ and the average number of generations the operator was used. The increment in the processing time for these operators was not significant.

The local search operator (Partial-Optimization) was tested, regarding its probability of application (p_{PO}) and the length of the intermediary segment

($POsize$) to be optimized. The range for these two parameters were: 1–8% and 5–8, respectively. The combinations of parameters lead to 32 experiments, done in the same way as before. Considering that the processing time tends to grow exponentially as the size of such parameters grow (in special, $POsize$), not only $avgHH$ should be taken into account, but also the processing time. Experimentally, we observed that when p_{PO} and $POsize$ were set both to 8%, the average processing time (using the previously mentioned standard parameters) raises to 660 s, when compared with 35.3 s with no Partial-Optimization. To meet both performance and processing time, we used the concept of Pareto optimality, commonly used in multiobjective problems [26]. Out from the Pareto front, we select $p_{PO} = 8\%$ and $POsize = 7$ amino acids as the operating point with better trade-off between the two conflicting objectives. At this point, the average processing time is only 122.1 s with a small decrement in performance, when compared with the combination of parameters that gives the better results.

4.2 Experiments with Synthetic Proteins

The first benchmark used is a set of synthetic amino acid chains, which optimal folding is known. Table 1 shows details of these chains that were used in several works in recent literature. In this table, NA and $maxHH$ stand, respectively, for the number of amino acids in the chain and the maximum number of H–H bonds, according to the 2D-HP model. This benchmark was presented by [27] (except the last instance that was introduced by [12]). The $maxHH$ values in the table are those originally given by authors, although for chains with 60 and 85 amino acids other results can be found in the literature.

Due to the stochastic nature of GA, all experiments described in this and the next sections were run for 100 times with different random seeds, and the averages are reported. Table 2 presents the results obtained by HGAPF together with those from [3] and [12] who also used a GA approach. More comparison of results is done with [28] that used PERM (Pruned Enriched Rosenbluth Method), [22] that used a variation of tabu search (GTabu),

Table 1. Benchmark of synthetic amino acids chains used in the experiments

NA	Amino acids chain	$maxHH$
20	$(HP)^2PH^2PHP^2HPH^2P^2HPH$	9
24	$H^2(P^2H)^7H$	9
25	$P^2HP^2(H^2P^4)^3H^2$	8
36	$P^3H^2P^2H^2P^5H^7P^2H^2P^4H^2P^2HP^2$	14
48	$P^2H(P^2H^2)^2P^5H^{10}P^6(H^2P^2)^2HP^2H^5$	23
50	$H^2(PH)^3PH^4PH(P^3H)^2P^4H(P^3H)^2PHPH^4(HP)^3H^2$	21
60	$P^2H^3PH^8P^3H^{10}PHP^3H^{12}P^4H^6PH^2PHP$	35
64	$H^{12}(PH)^2(P^2H^2)^2P^2HP^2H^2PPH^2P^2HP^2(H^2P^2)^2(HP)^2H^{12}$	42
85	$H^4P^4H^{12}P^6(H^{12}P^3)^3HP^2(H^2P^2)^2HPH$	52

Table 2. Comparison of results

NA	[27]	[12]		[28]	[22]	[29]	HGAPF	
	Best	Best	Mean	Best	Best	Best	Best	Mean
20	9	9(100)	9	9	9	9	9(74)	8.74
24	9	–	–	9	9	9	9(63)	8.61
25	8	–	–	8	8	8	8(69)	7.69
36	14	14(8)	12.40	14	14	14	14(6)	12.44
48	22	23(1)	18.50	23	22	23	23(2)	20.06
50	21	–	–	21	21	21	21(6)	18.72
60	34	–	–	36	35	35	35(6)	32.65
64	37	37(1)	29.30	42	42	39	40(5)	34.58
85	–	46(1)	40.80	53	50	52	51(2)	45.80

and [29] who employed an evolutionary Monte Carlo algorithm. Numbers within parenthesis indicate how many times the best score was found. It is important to highlight that the results from [3] represent the best individual taken out of five runs while those from HGAPF and [12] were run for 100 times. There is no information about repetibility for the other works. Comparing the results obtained for the four shortest chains and the 50 amino acids-long chain, it can be noticed that all algorithms have reached the maximum value of H–H bonds, despite some differences in the average. For the 48-residue chain, [27] and [22] did not find the best conformation while the others did. Considering the chain with 60 amino acids, [27] presented its result (34 contacts) as being the optimal, all the other algorithms found better conformations with 35 hydrophobic non-local bonds and PERM found another one with 36. For the two longest chains (64 and 85 amino acids), it can be observed that HGAPF found conformations better than the other GA implementations and, sometimes, better or worse than [22] and [29], but very close to the maximum known, found by [28].

At this point it is important to recall that the difference of a single H–H bond from a conformation to another indicates a great improvement obtained by the algorithm. Consequently, jumping from the closest local minimum to the global minimum can be considered a great achievement. For instance, the two best results found by HGAPF for the 85-residue chain are presented in Fig. 3. In the figure we can notice that both conformations are quite different, but have 51 hydrophobic non-local bonds. This example confirms the assertion mentioned before that the fitness hypersurface has many local maxima of comparable amplitudes. This fact, by itself, makes the task of finding the global optimum of the PFP very hard, for any algorithm.

4.3 Experiments with Real-World Proteins

A second set of experiments to evaluate the performance HGAPF was done using real-world proteins drawn from the Protein Data Bank – PDB [25].

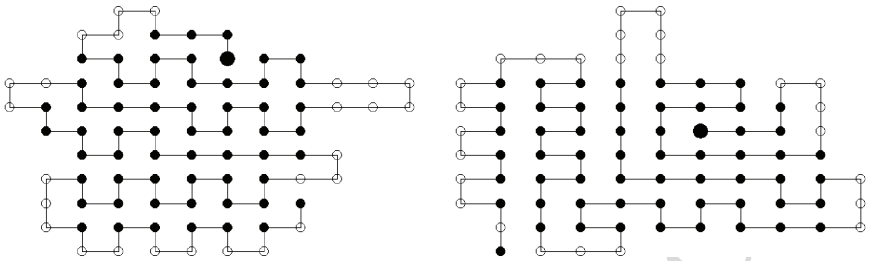

Fig. 3. Two best conformations for the 85 amino acids-long chain

PDB is a reference for Molecular Biology and all proteins in the database had their tertiary structure precisely determined by means of X-ray spectroscopy or similar methods. Therefore, data from PDB are considered reliable and the conformations of proteins are supposed to represent their real native states.

A PDB file has detailed information about structure, but it is necessary to "translate" a protein sequence, that uses the 20 standard amino acids, to a sequence of H's and P's, in order to use the HP model. A translation matrix, based on the amino acids chemical features, is used to define the hydrophilic (polar) or hydrophobic nature of standard amino acids. There are some divergences between authors (see, for instance, [23]) and the PDB, concerning to this translation matrix. Therefore, we decided to use the PDB definition of hydrophobic and hydrophilic amino acids, and we used this approach to construct the translation matrix for our experiments. Some real small proteins, ranging from 96 to 330 amino acids, were selected at random for building this benchmark, and their translation to the HP model, as well de PDB identification code, are presented in Table 3.

Since the maximum number of hydrophobic non-local bonds for these sequences is not known a priori, and they were not synthetically designed, it is not possible to evaluate precisely how good the solutions found are. Notwithstanding, a graphical analysis of the folding (not shown here) reveals that the best foldings reported here are not the global optimum for these sequences. However, hopefully these chains and results will be important for forthcoming researchers to improve their heuristic methods. Results are also shown in Table 3, where $maxHH$ stands for the maximum number of H–H contacts found by HGAPF with its standard parameters.

5 Discussion and Conclusions

This paper described an hybrid genetic algorithm for the protein folding problem using the 2D hydrophobic-polar (HP) model.

The use of the concept of radius of gyration in the fitness function is one of the main contributions of this approach. It takes smoothness to the fitness landscape, allowing better solutions to be found by forcing the hydrophobic

Table 3. Results for real-world proteins

PDB	NA	Protein sequence	$maxHH$
1mli	96	$H^{10}(HP)^3PH^4PH^3P^2HPH^2P^2HP^3H^3PH^3PH^6(PH)^2$ $(HP)^2(PH)^2H^2PH^3(PH)^2HPH^2(PH)^3H(HP)^2P^2H^2P^4$	46
256b	106	$(HP)^2P(PH)^2HP^3H^4(PH^2P)^2(H^2P)^2H^5PH^4P(HP)^2H^3$ $P^2H^2P^2HP^2(HP)^2P^2HPH^2P^3H^4PH^2P(PH^2)^2P^2H^5$P $H^4(PH)^2(H^2P^2)^2HPP$	48
1hmd	113	$H^2PHP^3H^3(PH)^3H^5P^3H^5PH^5(P^2H)^2PH^2P^2HP^2H^7$ $P^4H^2P^4H^5(PH^5P)^2H^2(PH)^2H^4(PH^3)^2H^2PH^3PH^3$	61
2ccy	128	$(P^3H)^2H^3P^2H^3PH^3P^2H^2PH^9(PH)^3(HP^2)^2H(H^7P)^2$ $H^2P^2HPH^2P^2H^3(PH)^3HPH^7P^2H^{10}P^2H^4PH^{12}P^2H^2P^2$	72
1d9i	288	$HPH^3P^2H^2PH^2PHP^2H^2P(P^2H^2)^2(PH)^2(HP)^2(PH)^2$ $H^2P^2HPH^4PHP^4H^5PH^2P^3H^{11}P(PH)^3HP^3H^3PH^4P^2$ $(HP)^2P(PH)^2HPH^2PH^5P^2(HP)^2P^2HP^3H^8PH^3P^2H^3P$ $H^3P^4H^3PH^2P(H^5P)^2H^2P(H^4P)^2P(PH^3)^2(PH)^2HP^3H^3$ $P^2(HP)^2H^2P^2H^7P^3H(HP)^2H^2PHP^2H^2PH^4P^2HP^3H^2$ $PH^4(PH^2)^2P^3H^{10}PH^5PH^3P^2H^2$	126
1epr	330	$(PH^2)^2H^2(PHP)^2PH^4(PH)^2H^2(PH)^2(HP)^3H^2P^3H^4P^3$ $H^3P(PH)^3H^3P(PH)^2H^5PH^4(PH)^2(HP)^2P^3HPH^3$ $(PH^2)^2H^6PH^2P^2H^4(P^3H^2)^2PH^7(PH^2)^2(P^2H)^2H^3P^2H^3$ $PHP^3H^4PH^5(PH^3)^2H(PH^6)^2(PH^2)^2H(PH^2)^2(H^4P)^2$ $(HP)^2H^3PH^4(H^4P)^3HPH^3P^4H^6(PH)^2H^2P^2H^6(PH)^2H^2$ $(PH)^2HPH^2(PH)^2HP^3H^5P^3H^2(H^3P)^2H^{10}PH^4PH^5PH$	160

residues to be placed in the core of the conformation. Using this fitness function improvement, two conformations with the same number of H–H bonds can be adequately discriminated.

Another important enhancement is the Partial-Optimization operator, which performs a local search in a part of the individual and, most times, improves the overall conformation. Despite its utility, it tends to be computationally expensive as the number of intermediate residues increases, and should be used with parsimony.

The Improved-Mutation operator also added significant improvement to the GA, working together with the regular genetic operators. This special operator is especially useful in the last generation when it is applied to all individuals of the population hopefully improving further the solutions found. The U-Fold and Make-Loops operators have also contributed to the interesting results obtained.

The Decimation-and-Hot-Boot strategy was necessary because of the many local maxima in the fitness landscape. This strategy allows the algorithm to restart without loosing useful information (represented by the best individual found up to that moment). DHB does not improve individuals, but gives conditions to the GA to recover from the effects of a premature convergence, possibly allowing a more efficient exploration of the search space.

Genetic algorithms have been applied to many problems and, as a matter of fact, the more complex and complicated a problem is, the more sophisticated a GA have to be. As a consequence, a GA implementation will have many parameters to be tuned for the specific class of problems. Therefore, such kind of application will require a previous analysis of parameters' sensitivity. Our analysis of the GA parameters aimed at suggesting a set of values to the parameters. Those values are expected to make the GA perform well for many instances of the problem. Several tests were done to evaluate the influence of each parameter over the evolution process. For the parameters related to improved mutation and partial optimization operators, a different analysis was done because they are very time costly. The analysis of the Pareto front identifies not a single value, but a set of values that are equally good, considering both conflicting objectives: performance maximization and processing time minimization. Users will choose specific sets of parameter values according to specific situations, deciding what is more important at the moment, better results or a shorter processing time. It is worth to recall that the more sophisticated a GA, the more difficult is to adjust its running parameters for a given problem. This is where methods for self-adjustment of parameters [30] may be a valuable aid to the user.

It is important to emphasize the results obtained over the case studies. The first data set was taken from the literature and comprises synthetically designed amino acids chains with different lengths whose optimal folding is supposed to be known. Most algorithms in the literature perform well for short chains. However, for the longer ones, different local maxima were found, suggesting that there is room for further testing with longer chains. Our proposed GA performed better or, at least, similarly to other algorithms found in the literature.

The second data set was composed by six real-world proteins selected from PDB and translated to the HP model according to the PDB definitions. After some non-exhaustive runs, the best conformation found for each sequence was presented. The maximum number of H–H bonds presented for these proteins is certainly far from the optimal unknown value, but, even so, they represent good foldings. A graphical analysis of the conformations found reveals that, our GA was able to group most of the hydrophobic residues conveniently in the core of the protein, pushing hydrophilic residues away to peripheral regions. We can also notice that in real proteins there are some hydrophilic residues isolated in the sequence, what makes this kind of sequence very complicated to be folded. As we mentioned before, due to the nature of the fitness landscape, small improvements (i.e., increasing the number of H–H bonds by few units) in a given conformation are very hard to obtain.

Let us consider that the folding process takes several steps, from an unfolded sequence to its minimal energy conformation. It is not difficult to imagine that, the more folded the chain, the more difficult is to find one or more changes in the structure that can lead to a better conformation. Therefore, from the computational point of view, the folding is a problem

of increasing difficulty. Hence, any heuristic algorithm devised to solve this problem must be powerful enough to cope with this situation. The hybrid GA approach proposed in this work has some limitations, especially when dealing with long chains. This fact suggests the necessity of even more intelligent operators that can analyze regions of the conformation and make context-dependent changes. Possibly, such improvements will lead to computationally intensive local search strategies, and this is an issue to be addressed in a future work.

The 2D-HP model is very popular, but it is far away from reality. More realistic models are needed, despite its computational complexity. In the same way, heuristic methods will play an important role to offer solutions for the protein folding problem. Also, more powerful computational resources certainly will be required for dealing with real-world instances of the problem. This is a current trend in the area, since researchers are moving from simple workstations to hardware-based reconfigurable computing [31] and grid computing [32]. Overall, results of this work encourage the continuity of the research, since there is still no efficient method for solving large instances of the PFP. Particularly, future developments will be towards a more complex lattice model and improved genetic operators.

Acknowledgements

The authors gratefully acknowledge the Brazilian National Research Council (CNPq) for the financial support to H.S. Lopes (grant 305720/04-0) and M.P. Scapin (grant 131355/04-0).

References

1. Berger B, Leight T (1998) Protein folding in the hydrophobic–hydrophilic (HP) model is NP-complete. J Comput Biol 5:27–40
2. Unger R, Moult J (1993) On the applicability of genetic algorithms to protein folding. In: Proc. of IEEE 26th Hawaii Int. Conf. on System Sciences, vol. I, pp. 715–725
3. Unger R, Moult J (1993) A genetic algorithm for three dimensional protein folding simulations. In: Proc. of the 5th Annual Int. Conf. on Genetic Algorithms, pp. 581–588
4. Patton AL, Punch III WF, Goodman ED (1995) A standard GA approach to native protein conformation prediction. In: Proc. of the 6th Int. Conf. on Genetic Algorithms, pp. 574–581
5. Pedersen JT, Moult J (1997) Protein folding simulations with genetic algorithms and a detailed molecular description. J Mol Biol 269:240–259
6. Krasnogor N, Pelta D, Lopez PEM, Canal E (1998) Genetic algorithm for the protein folding problem: a critical view. In: Proc. of Engineering of Intelligent Systems, pp. 353–360

7. Day RO, Lamont GB, Pachter R (2003) Protein structure prediction by applying an evolutionary algorithm. In: Proc. of Int. Parallel and Distributed Processing Symp., pp. 155–162
8. Bitello R, Lopes HS (2006) A differential evolution approach for protein folding. In: Proc. IEEE Symp. on Computational Intelligence in Bioinformatics and Computational Biology, pp. 1–5
9. Shmygelska A, Hoos HH (2005) An ant colony optimisation algorithm for the 2D and 3D hydrophbic polar protein folding problem. BMC Bioinformatics 6:30–52
10. Dill KA (1985) Theory for the folding and stability of globular proteins. Biochemistry 24:1501–1509
11. Jiang T, Cui Q, Shi G, Ma S (2003) Protein folding simulations of the hydrophobic–hydrophilic model by combining tabu search with genetic algorithms. J Chem Phys 119:4592–4596
12. König R, Dandekar T (1999) Improving genetic algorithms for protein folding simulations by systematic crossover. Biosystems 50:17–25
13. Dill KA, Bromberg S, Yue K, Fiebig KM, Yee DP, Thomas PD, Chan HS (1995) Principles of protein folding – a perspective from simple exact models. Protein Sci 4:561–602
14. Crescenzi P, Goldman D, Papadimitriou C, Piccolboni A, Yannakakis M (1998) On the complexity of protein folding. J Comput Biol 5:423–465
15. Michalewicz Z (1996) Genetic Algorithms + Data Structures = Evolution Programs, 3rd edition. Springer, Berlin
16. Piccolboni A, Mauri G (1998) Application of evolutionary algorithms to protein folding prediction. Lect Notes Comput Sci 1363:123–136
17. Krasnogor N, Hart WE, Smith J, Pelta DA (1999) Protein structure prediction with evolutionary algorithms. In: Proc. Int. Genetic and Evolutionary Computation Conf., pp. 1596–1601
18. Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT, Cambridge
19. Beer FP, Johnston ER (1980) Vector Mechanics for Engineers – Statics. McGraw Hill, New York
20. Cox GA, Mortimer-Jones TV, Taylor RP, Johnston RL (2004) Development and optimisation of a novel genetic algorithm for studying model protein folding. Theor Chem Acc 112:163–178
21. Hoque MT, Chetty M, Dooley LS (2006) A guided genetic algorithm for protein folding prediction using 3D hydrophobic–hydrophilic model. In: Proc. IEEE Congr. on Evolutionary Computation, pp. 2339–2346
22. Lesh N, Mitzenmacher M, Whitesides S (2003) A complete and effective move set for simple protein folding. In: Proc. 7th Ann. Int. Conf. Research in Computational Molecular Biology, pp. 188–195
23. Lehninger AL, Nelson DL, Cox MM (1998) Principles of Biochemistry, 2nd edition. Worth, New York
24. Croes GA (1958) A method for solving traveling salesman problems. Oper Res 6:791–812
25. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28:235–242
26. Deb K (2000) Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester
27. Unger R, Moult J (1993) Genetic algorithms for protein folding simulations. J Mol Biol 231:75–81

28. Chikenji G, Kikuchi M, Iba Y (1999) Multi-self-overlap ensemble for protein folding: ground state search and thermodynamics. Phys Rev Lett 83:1886–1889
29. Liang F, Wong WH (2001) Evolutionary Monte Carlo for protein folding simulations. J Chem Phys 115:3374–3380
30. Maruo MH, Lopes HS, Delgado MRBS (2005) Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. Lect Notes Comput Sci 3448:154–165
31. Armstrong Jr. NB, Lopes HS, Lima CRE (2007) Reconfigurable Computing for Accelerating Protein Folding Simulations. Lect Notes Comput Sci 4419:314–32
32. Tantar A-A, Melab N, Talbi E-G, Parent B, Horvath D (2007) A parallel hybrid genetic algorithm for protein structure prediction on the grid. Future Gen Comput Syst 23(3):398–409