Diversity Control in Genetic Algorithms for Protein Structure Prediction

Vinicius Tragante do Ó[1], Renato Tinós[1]

[1] Departamento de Física e Matemática - FFCLRP, Universidade de São Paulo
Av. Bandeirantes, 3900, Ribeirão Preto - SP, Brasil
tragante@pg.ffclrp.usp.br, rtinos@ffclrp.usp.br

Abstract. *In recent years, there is a growing interest in using Genetic Algorithms (GAs) in the protein structure prediction problem. However, the search space in this problem is very complex, what results in premature convergence of the GAs in their standard form, as the population generally gets trapped into local optima. Based on this fact, the use of two different strategies that can help GAs to maintain or increase the diversity of the population in the protein structure prediction problem are investigated in this paper. These strategies are Hypermutation and Random Immigrants. A new form of codification of the protein structure in the GA using sorted angles database is still proposed. Experimental results with Crambin (PDB code 1CRN), Met-Enkephalin (PDB code 1PLW), and DNA-Ligand (PDB code 1ENH) show that strategies to increase or maintain the population diversity are interesting for the protein structure prediction problem.*

Keywords: *Genetic Algorithms, Hypermutation, Random Immigrants, Protein Structure Prediction.*

1 Introduction

Proteins are polypeptide chains composed of a sequence of any of the 20 existing amino acids. In Biochemistry, the tertiary structure of a protein is its three-dimensional shape, also known as its fold [Lehninger, 2005]. The function of a protein is intimately related to its folding, which means that its tertiary structure determines what a protein is capable of doing, mainly because of its hydrophobic/hydrophilic cores, hydrogen bonds, and interactions with other molecules.

Protein structure prediction is one of the most important unsolved problems in molecular biology. It is already known that the activation or inhibition of molecules is dependent on the docking between an activator/inhibitor and a target molecule. Based on this fact, it is possible, in the future, to develop specific molecular structures capable of acting over toxic agents using protein structure prediction, allowing the development of new drugs, which may help solving a huge amount of health problems for which there is no efficient treatment yet.

Nowadays, the best known methods to determine an existing protein's tertiary structure are crystallography and nuclear magnetic resonance, but both methods are either expensive or have limitations. Ideally, it would be possible to determine a protein structure based only on its amino acids sequence; this, however, is not possible yet for a large range of proteins with a medium or large number of amino acids [Lehninger, 2005].

Several approaches for the protein structure problem have been investigated since the 1950's, some of them known as molecular dynamics, homology modeling, and *ab initio*. Genetic Algorithms (GAs) seem to be suitable for the protein folding problem, mainly in the *ab initio* approach, since this problem can be viewed as a search problem, in which, given an amino acid sequence, the best structure amongst all must be found. GAs are particularly attractive to this problem due to its characteristics [Mitchell, 1996].

In a GA, a population of chromosomes, representing a series of candidate solutions (called individuals) to an optimization problem, generally evolves toward better solutions. The evolution usually starts from a population of randomly generated individuals. In each generation, the fitness of every individual is evaluated, the best individuals are selected (elitism), and the rest of the new population is formed by the recombination of pairs of individuals, submitted to random mutations. The new population is then used in the next generation of the algorithm. Commonly, as employed in this problem, the algorithm ends when a maximum number of generations is reached.

Indeed, GAs have been successfully applied to several areas where optimization is a requisite, such as attribute selection [Yang & Honavar, 1998], logistics [Taniguchi *et al.*, 1998], electrical systems [Fukuyama *et al.*, 1998], among others. In the protein folding problem, GAs in the standard configuration have been applied, but without major success, mainly because of the existence of several local optima over the search space and the use of imprecise energy function to be minimized. In fact, the choice of the energy function has a relevant impact in the optimization process. The energy function is difficult to be implemented with sufficient precision because, among others, there are too many interactions among a large number of atoms. This fact was clear in [Schulze-Kremer, 1993], a pioneer work, where GAs were applied to the protein structure prediction problem. In [Schulze-Kremer, 1993], a small set of angles was used as a set of possible solutions for each torsion angle of each amino acid of the protein, and the GA reached even lower energy levels than the protein in its native state; however, this was not enough to determine the real native state. Today, computing power has increased significantly, which allows us to use bigger angle database sets, more complete force fields, and improve the GA with new strategies in order to reach better results [Gabriel *et al.*, 2007].

In this work, the effects of two GA' strategies, generally applied to dynamic optimization problems [Cobb & Grefenstette, 1993] to increase or maintain the diversity of the population, were investigated in the protein structure prediction. The strategies are Hypermutation, which raises the mutation rates periodically, and Random Immigrants, which replaces a percentage of the individuals each generation by new, randomly generated ones. Both strategies help keep the diversity of the population, and can allow the GA to avoid the premature convergence around one local optimum. The solutions of the GAs were codified using databases of torsion angles found in proteins which tertiary structure determined by crystallography or nuclear magnetic resonance. The use of sorted databases is proposed in this paper.

In Section 2, the methodology developed in this work is presented. Section 3 shows the experimental results obtained with the proteins *Crambin (PDB code 1CRN), Met-Enkephalin (PDB code 1PLW), and DNA-Ligand (PDB code 1ENH)* for different GA's approaches. Section 4 presents the conclusions.

2 Methods

Studies found in literature state that the number of possible combinations for the torsion angles makes the protein structure prediction an *NP*-hard problem [Pierce & Winfree, 2002]. However, not all combinations have minimum energy levels (or are frequent combinations), as shows the Ramachandran plot [Ramachandran & Sasiekharan, 1968], what results in a smaller search space.

Based on this fact, new approaches have included angle databases composed of fixed values of torsion angles *phi* (ϕ) and *psi* (ψ) for the main chain and angles of the side chain (see Fig. 1 for a graphical demonstration of these angles), which may consist of angles $\chi 1$ to $\chi 5$ depending on the amino acid. Such databases are composed of angles, experimentally determined by nuclear magnetic resonance or crystallography, of the amino acids found in hundreds of proteins. With this in mind, this work used a set of these angles recorded into an angle database by the project CADB (*Conformational Angle Database*) [Sheik *et al.*, 2003]. The side chain also has its own database, which is based on the Tuffery Database [Tuffery *et al.*, 2003] (found in http://bioserv.rpbs.jussieu.fr/doc/Rotamers.html). This database was created starting from observations of protein structures determined with magnetic resonance and crystallography, and ordered by the frequency of appearance in these structures.

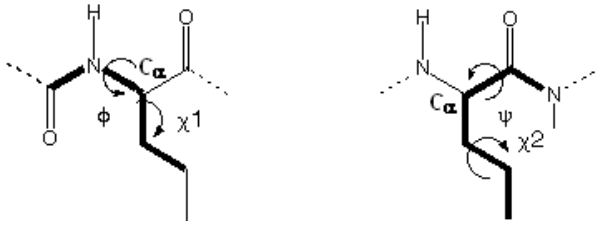

Fig. 1. Angles used as inputs for the chromosome of each individual of the GA. Starting from C_α, which is the base carbon, we see ϕ and ψ angles which are the torsion angles from the main chain; and χ_1 and χ_2 angles, which are the angles from the side chain; depending on the amino acid, there can possibly be up to 5 χ angles.

In this work, the chromosome of each individual is formed by the index of the database of each amino acid and the index of the side chain database, as shown in Fig. 2, which means that the chromosome size is *2m*, where *m* is the size of the protein (number of amino acids). All angle values are saved into an extra vector, which is faster than searching the database sets each time a value is needed. In this work, we propose to sort the main chain angles databases according to the torsion angle ϕ (when the angle ϕ is equal for two or more combinations, the angle ψ is also sorted). In Section 3, experimental results with sorted and unsorted databases are presented.

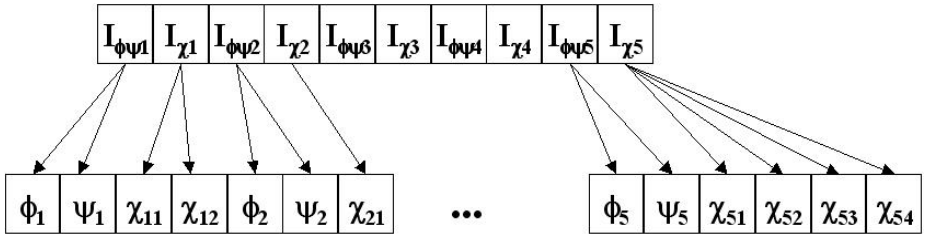

Fig. 2. Graphical schema of a chromosome for a protein with 5 amino acids. Each amino acid is represented by two values, I$\phi\psi$ and Iχ, which are the indexes of the database set for the main chain and the side chain. An auxiliary vector stores the real values of the angles.

In this work, the mutation rate of the GA was set to *1/(2m)* and the crossover rate was set to 0.8. When a gene (database index position) is mutated, the index position is changed in one position, i.e., +1 or –1 is added to the parent index position. When a mutation happens, all torsion angles of that amino acid are generally changed (either from the main chain or the side chain). In this way, when the main chain databases are unsorted, a simple mutation very often generates a new gene (of the offspring) with angles far from the angles given by the old gene (of the parent). When the main chain databases are sorted, the new gene is generally generated close to the old gene if mutated, as the new torsion angle ϕ (offspring) has a value close to the old torsion angle ϕ (parent).

The crossover uses 2 parents that are chosen by tournament selection, in which 2 individuals are picked randomly and the individual with the best fitness between them has 75% of being chosen for crossover, with 25% remaining for the individual with worst fitness [Mitchell, 1996]. The crossover method chosen is the single-point crossover, in which a random point is chosen, and the left side of this point is obtained from the first individual chosen and the right side of this point, from the second individual to generate the offspring 1, and the inverse method for the offspring 2 (Figure 3). We inserted also an elitist process, by finding the two best individuals from the generation and automatically inserting them in the next population, without crossover or mutation. This procedure was used in the Schulze-Kremer approach [Schulze-Kremer, 1993] and in [Gabriel *et al.*, 2007], [Cui *et al.*, 1998].

When the individuals are completely defined, the GA creates a file that contains the ϕ, ψ, ω, and χ angles of each amino acid and sends it to the algorithm *protein* from the molecular modeling package Tinker [Ponder *et al.*, 1998], which can convert these torsion angles into a pdb or a xyz file. The pdb file is the type of representation found in the PDB database (http://www.rcsb.org/), while the xyz file is a representation of each atom of the protein in its coordinates in space. This step is necessary to evaluate the fitness of each candidate solution (individual), in the algorithm *analyze*, also a component of the molecular modeling package Tinker. This program uses the file generated by the *protein* algorithm to verify the interactions that are present in the protein and informs the total energy that each individual has. The purpose of the GA is to reduce this energy to the minimum possible value.

The evaluation of the Tinker package depends on the force field chosen. In this work, the CHARMM27 force field was used. The total energy (fitness) is given by:

$$E_{tot} = E_{bs} + E_{ab} + E_{id} + E_{ta} + E_{VdW} + E_{cc} \quad (1)$$

where: E_{bs} is the bond stretching energy, which measures the energy according to the distance of the bonding; E_{ab} is the angle bending energy; E_{UB} is the Urey-Bradley energy; E_{it} is the improper dihedral energy, which is associated to the deformations of improper torsion angles; E_{ta} is the torsional angle energy; E_{VdW} is the Van der Waals

energy; and E_{cc} is the charge-charge energy, which is represented by the Coulomb potential.

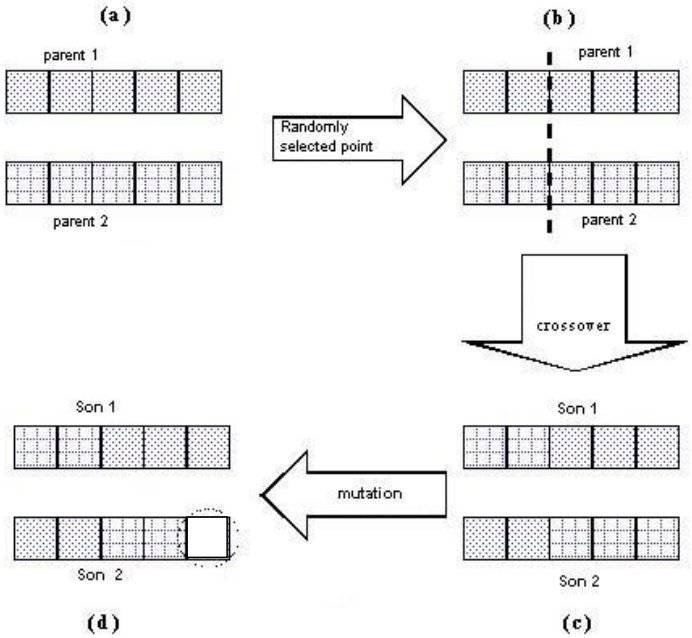

Fig. 3. The two selected parents for crossover (a). After the selection of the point of separation (b), a part from parent 1 and other part from parent 2 are recombined to generate offspring (c) and mutation operator may alter one or both individuals (d). Image adapted from [Linden, 2006].

In order to maintain or increase the diversity of the population of the GA over the generations, two approaches called Hypermutation and Random Immigrants were tested. Hypermutation, first described by Cobb & Grefenstette [Cobb & Grefenstette, 1993], is a strategy that increases the mutation rate according to a given criterion; for example, when the fitness of the best individual decreases in dynamic optimization problems. Another approach for Hypermutation, which was used in this work, increases the mutation rate in 5 generations in a row, then returns to standard mutation rate for the following 5 generations, during the whole execution of the algorithm. Other numbers of generations could have been used, but this one worked well in this problem for keeping the diversity among individuals. The following pseudo-code illustrates this idea.

Pseudo-code for the Hypermutation procedure

```
procedure generation ()
  begin
    father1 = tournament()      //selection of father 1
    father2 = tournament()      //selection of father 2
    son=father1.crossover(father2) //crossover
    if (flag==0)
      mutation_rate = normal_rate
      counter++
      if (counter=5)
        flag=1
        counter=0
        break
      end_if
    else
      mutation_rate = high_rate
      counter++
      if (counter=5)
        flag=0
        counter=0
      end_if
    end_if
    son[0].mutation(mutation_rate)
    son[1].mutation(mutation_rate)
  end.
```

On the other hand, the Random Immigrants approach [Cobb & Grefenstette, 1993], [Vavak & Fogarty, 1996] replaces a percentage of individuals from the population defined by a replacement rate by new, randomly generated ones. In this work, random individuals are automatically inserted in the next population every generation. The pseudo-code below describes this procedure.

Pseudo-code for the Random Immigrants procedure

```
procedure generation ()
  begin
    while (total_immigrants < percentage)
      son=new_individual()
      new_population.add(son)
    total_immigrants++
  end_while
    for(counter=percentage;counter<pop_size;counter+2)
      father1 = tournament()      //selection of father 1
      father2 = tournament()      //selection of father 2
      son=father1.crossover(father2) //crossover
      son[0].mutation(mutation_rate)
      son[1].mutation(mutation_rate)
    end_for
end.
```

These procedures, as we will see in next section, maintain or increase the diversity of the population, helping the algorithm to avoid the premature convergence to one local optimum.

3 Results

The algorithms were tested using three proteins obtained from the PDB bank (http://www.rscb.org/pdb): Met-Enkephalin (PDB code 1PLW), Crambin (PDB code 1CRN), and DNA-Ligand (PDB code 1ENH). In the Charmm27 force field, the parameter "dielectric" was changed to 78.7, in order to simulate the effect of the presence of water around the protein. In the first generation of the GAs, individuals were randomly initialized by picking a random position of the database for each amino acid of each individual. The process was repeated *n* times, being *n* the number of individuals per generation. For proteins Crambin and DNA-Ligand, which are respectively 46 and 55 amino acid long and have been used in other works [Gabriel *et al.*, 2007] [Pedersen & Moult, 1996], 500 generations of 100 individuals were evolved

in order to reach the final result. Some experiments with 1000 generations indicated that there is little improvement in the performance after 500 generations. For protein Met-Enkephalin, which is 5 amino acids long and has also been used in other works [Nicosia & Stracquadanio, 2008] [Bindewald *et al.*, 1998] [Kaiser *et al.*, 1997], 50 generations of 100 individuals were evolved. Each algorithm was executed for ten different random seeds, one per run, but the seeds were the same for all approaches. The size of the population, 100 (which is a standard value for GAs), remains fix along the generations since mutations and immigrants replace a fraction of the population, keeping constant the amount of individuals.

The three GA approaches (Standard, Hypermutation, and Random Immigrants) were tested using both sorted and unsorted database sets. The database sets were sorted first by the ϕ angle, from –180º to 180º; in case of two equal ϕ angles, then ψ angle was used for sorting. When the Random Immigrants approach was used, four replacement rates (2%, 6%, 10%, and 30% of the population per generation) were tested.

3.1 Crambin

Table 1 shows the results of best fitness, population mean fitness, and standard deviation obtained for protein Crambin. The presented results were averaged over 10 executions. The value of fitness for the native state of protein Crambin is also presented.

Table 1. Results for protein Crambin.

Algorithm	Best Fitness	Average Fitness	Std. Deviation
Hypermutation (sort)	586.178	716.465	88.462
Hypermutation (unsort)	581.893	672.237	87.112
Random Im. 2% (sort)	561.596	574.746	11,978
Random Im. 2% (unsort)	559.022	590.557	30.941
Random Im. 6% (sort)	506.252	525.767	16.054
Random Im. 6%(unsort)	517.040	538.819	11.936
Random Im. 10% (sort)	519.987	538.219	18.476
Random Im. 30%(sort)	661.803	735.586	43.491
Standard GA (sort)	695.754	831.733	110.018
Standard GA (unsort)	626.908	816.237	247.777
Native state	465.538		-

From the results, it is possible to notice that both Hypermutation and Random Immigrants reach better results than the Standard GA. Indeed, Student's T tests were performed, and when comparing Hypermutation using the sorted database against the Standard GA (also with sorted database), the score was 0.019, which means a strong probability that the results are really different (not due to sampling errors). For the unsorted database for both methods, the score was 0.11, which is not so strong, but shows that there is little probability that this difference was due to sampling errors. When comparing the Random Immigrants approach with 6% of replacement of individuals per generation (which was the best replacement rate found), the probability was under 10^{-6} using the sorted database sets, and under 10^{-3} for the unsorted approach, which indicates that this approach was better than the Standard GA for this problem.

3.2 Met-Enkephalin

Table 2 shows the results for Met-Enkephalin. One can observe that all approaches were able to find a lower potential energy than the real native state, a result that was already reported in the literature (see Section 1) [Pedersen & Moult, 1996].

Table 2. Results for protein Met-Enkephalin (50 generations).

Algorithm	Best Fitness	Average Fitness	Std. Deviation
Hypermutation (sort)	43.736	46.237	1.50
Hypermutation (unsort)	44.492	46.797	1.078
Random Im. 2% (sort)	43.420	46.577	1.284
Random Im. 2% (unsort)	44.602	46.618	0.899
Random Im. 6% (sort)	44.86	46.439	0.979
Random Im. 6% (unsort)	43.404	45.737	1.246
Random Im. 10% (sort)	44.847	46.160	0.848
Random Im. 30%(sort)	46.426	47.907	0.670
Standard GA (sort)	45.599	47.107	1.092
Standard GA (unsort)	46.203	47.598	1.223
Native State	345.978		-

Analyzing the results, it is possible to observe that, again, all approaches were better than the Standard GA. When comparing Hypermutation and the Standard GA, there was a probability of only nearly 15% of sampling error in the T test, but best individuals were found for Hypermutation, which suggests better performance. For the Random Immigrants approach, once again the best replacement rate was 6% for this problem (in this way, the results for the unsorted database with 10% and 30% are not presented), and the score of the statistical comparison was under 5% when the Random Immigrants with 6% was compared to the Standard GA.

3.3 DNA-Ligand

The biggest protein tested (55 amino acids long) is also the most computationally costly among the proteins in this section. Table 3 presents the results for the DNA-Ligand.

Table 3. Results for protein DNA-Ligand.

Algorithm	Best Fitness	Average Fitness	Std. Deviation
Hypermutation (sort)	1018.911	4920.488	4226.027
Hypermutation (unsort)	1053.500	2073.168	986.010
Random Im. 2% (sort)	795.085	1047.238	183.626
Random Im. 2% (unsort)	704.036	1196.289	458.129
Random Im. 6% (sort)	691.593	713.582	12.922
Random Im. 6% (unsort)	673.558	728.646	60.821
Random Im. 10% (sort)	746.979	868.154	101.005
Standard GA (sort)	1446.176	3721.321	2794.986
Standard GA (unsort)	1077.668	4290.645	5047.524
Native State	427.305		-

Random Immigrants with replacement rate equal to 30% (sorted and unsorted) and 10% (unsorted) were not executed for DNA-Ligand. Once again, the approach with 6% of replacement rate reached the best results among all tested algorithms, and except for the Hypermutation with the sorted database, all algorithms were statistically better than the Standard GA, with under 5% probability of sampling error.

4 Analysis and Concluding Remarks

In this paper, we studied the effects of inserting diversity control strategies in GAs and the use of sorted angles databases applied to the problem of protein structure prediction.

In problems with a large solutions' space, it is difficult for the GA to extend the search over many spots because of its characteristics of converging the population to local optima. Because of this, it is useful to use mechanisms to maintain the diversity of the population, and to minimize the premature convergence problem.

For the proteins studied here, both strategies, Hypermutation and Random Immigrants, were capable of reaching better performance than the Standard GA, specially using a replacement rate of 6% for new immigrants. This rate is ideal because it is not so small that this diversity cannot be assimilated by the population, as the replacement rate of 2% shows us, and it is also not so big that there is no convergence, since individuals are replaced too fast for their best characteristics to be kept, which is the case of the 30% replacement rate. However, the fitness function has proved not to be very suitable, because when the original structures obtained from PDB and the structures created by the GAs were compared, there was low similarity. So, more interactions should be considered in modeling the fitness function, for example the hydrophilic/hydrophobic interactions with other atoms. Nonetheless, this does not alter the fact that maintaining or increasing the diversity on the population helps the algorithm to reach better results, what can be generalized if fitness function with more details is used, as the presence of a large amount of local optima is an intrinsic characteristic of the protein prediction problem. It is important to observe that the diversity produced by the introduction of random individuals (Random Immigrants) or by increasing mutation rates (Hypermutation) helps the population escape local optima.

Also, the use of sorted and unsorted databases can produce difference on the overall performance. In general, the use of unsorted databases resulted, in some cases, in a better performance, what can be explained because the mutation of only one gene eventually caused a great change in the representation of the mutated gene (amino acid), allowing a large jump in the search space which was beneficial. However, it was observed that more improvements in the best-of-generation fitness was reached when sorted databases were employed, what can be explained because, in this case, a mutation causes a small change in the angles of the amino acid, what can result in a deeper exploration of the current best solution neighborhood, as shown in [Tragante & Tinós, 2008]. In this way, if more generations are considered for the problem, the use of sorted databases should be more interesting.

In the future, new approaches for the Random Immigrants should be investigated in order to make the population replacement rates a dynamic process. New schemes to increase the population diversity should still be tested. Also, the use of GAs should be compared to other known methods in the protein structure prediction problem.

Acknowledgments. The authors thank Fapesp (process 04/04289-6) and CAPES for the financial support to this project.

References

Bindewald, E., Hesser, J., Manner, R. (1998) "Implementing genetic algorithms with sterical constrains for protein structure prediction", In *Proceedings of International Conference on Parallel Problem Solving from Nature (PPSN V)*, 959–967, Amsterdam, Netherlands.

Cobb, H.G., Grefenstette, J.J. (1993) “Genetic algorithms for tracking changing environments”, In *5th International Conference on Genetic Algorithms*, Morgan Kaufmann, 523-530.

Cui, Y., Chen, R., Wong, W. (1998) “Protein Folding Simulation With Genetic Algorithm and Supersecondary Structure Constraints”, PROTEINS: Structure, Function, and Genetics, 31:247–257.

Fukuyama, Y., Chiang, H., Miu, K. (1996) “Parallel genetic algorithm for service restoration in electric power distribution systems”, International Journal of Electrical Power and Energy Systems, 18(2):111–119.

Gabriel, P., Lima, T., Delbem, A., Faccioli, R., Silva, I. (2007) “Pure ab initio evolutionary approach to protein structure prediction”, In *International Symposium on Mathematical and Computation Biology*. BIOMAT 2007.

Kaiser Jr., C.E., Lamont, G.B., Merkle, L.D., Gates Jr., G.H., Patcher, R. (1997) “Polypeptide structure prediction: Real-valued versus binary hybrid genetic algorithms”, In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 279–286, San Jose, CA.

Lehninger, A.L., Nelson, D.L., Cox, M.M. (2005) Principles of Biochemistry 4 ed., Freeman, New York.

Linden, R. (2006) Algoritmos Genéticos. Ed. Brasport, Brasil.

M. Mitchell. (1996) An Introduction to Genetic Algorithms. Bradford Books.

Nicosia, G., Stracquadanio, G. (2008) “Generalized Pattern Search Algorithm for Peptide Structure Prediction”, Biophysical Journal 95(10):4988 - 4999.

Pedersen, J., Moult, J. (1996) “Genetic algorithms for protein structure prediction”, Current Opinion in Structural Biology, 6(2):227–231.

Pierce, N.A., Winfree, E. (2002) “Protein Design is NP-hard”, Protein Engineering, 15 (10):779-782.

Ponder, J. *et al.* (1998) TINKER: Software Tools for Molecular Design. Department of Biochemistry and Molecular Biophysics, Washington University School of Medicine, St. Louis, MO.

Ramachandran, G.N., Sasisekharan, V. (1968) “Conformation of polypeptides and proteins”, Advances in Protein Chemistry 23:283-438.

Schulze-Kremer, S. (1993) “Genetic Algorithms for Protein Tertiary Structure Prediction”, Lecture Notes in Computer Science: Machine Learning: ECML-93, 262-279.

Sheik, S.S.; Ananthalakshmi, P.; Bhargavi, G.R.; Sekar, K. (2003) “CADB: Conformation Angles DataBase of proteins”, Nucleic Acids Research 31(1):448-451.

Taniguchi, E., Noritake, M., Yamada, T., Izumitani, T. (1999) “Optimal size and location planning of public logistics terminals”, Transportation Research Part E, 35(3):207–222.

Tragante, V., Tinós, R. (2008) "Impact of Database Sorting on the Efficiency of Genetic Algorithms in Protein Structure Prediction", In *International Symposium on Mathematical and Computation Biology (BIOMAT'2008)*, Campos do Jordão, Brazil.

Tuffery, P., Etchebest, C., Hazout, S., Lavery, R. (1991) "A new approach to the rapid determination of protein side chain conformations", Journal of Biomolecular Structure & Dynamics, 8(6):1267–89.

Vavak, F., Fogarty, T.C. (1996) "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments", In *AISB Workshop on Evolutionary Computing*, Lecture Notes in Computer Science, Springer, 1143:297–304.

Yang, J., Honavar V. (1998) "Feature subset selection using a genetic algorithm", In: *Feature Extraction, Construction and Selection: a data mining perspective*, 117-136. Kluwer.