Chapter 12

Evolutionary Computation Techniques for Two Computational Biology Problems

Carlos A. Brizuela-Rodríguez, Milton Rodríguez-Zambrano and Jorge E. Luna-Taylor

Centro de Investigación Científica y de Educación Superior de Ensenada,
Km. 107 Carr. Tijuana-Ensenada, Ensenada, B.C.
Email: {cbrizuel, milton, lunat}@cicese.mx

Abstract

This chapter deals with the design of evolutionary algorithms for two well known computational biology problems: whole-genome shotgun assembly and a simplified model of the protein folding. The folding problem is one of most challenging open problems in biology, and any clue on how to solve it or its approximations will be very valuable. Evolutionary algorithms have shown their suitability for complex optimization problems. The idea to solve the problems is to use a genetic algorithm tailored to specific problem knowledge. The proposed method has proven to be an effective alternative to solve both problems.

Keywords: Shotgun Assembly, Protein Folding, Genetic Algorithms, Monte Carlo, Local Search.

1 Introduction

In this chapter we present two very important computational biology problems: DNA assembly and an approximation to the protein folding. The latter constitutes one of the most challenging open problems in molecular biology.

DNA sequencing is the process to decipher the precise order of bases along an unknown DNA chain. With today's technology it is possible to sequence 300 to 1000 bp[1] [1], in a single experiment. However, whole genomes are many times longer, for instance the human genome, without being the largest is in the order of 10^9 bp. To overcome this situation the whole-genome shotgun sequencing has been developed. This technique developed by Sanger and Coulson [2] was applied to sequence the *lambda* bacteriophage.

The shotgun method consists on randomly fragmenting the original DNA sequence. Each of the resulting fragments (reads) are then sequenced, and finally an assembly stage takes place, long fragments or contigs of the original sequence are reconstructed.

Many strategies have been proposed to deal with this problem, among these, greedy methods have been the most widely used. Here we propose a genetic algorithm to deal with the assembly problem, motivated mainly by the success on the application of this method to a wide class of combinatorial optimization problems, category where the assembly problem belongs.

There are many available programs to deal with this problem, among them we can cite: PHRAP[2], TGI[3], CAP3 [3], ARACHNE [4] and EULER[4] [5], the first four are based on greedy algorithms while the last one abandon what is known as the Overlap-Layout-Consensus (OLC) approach and is based on the D'brujin graph representation where the goal is to find a super Eulerian path.

Previous attempts to solve the problem by evolutionary computation have been the proposal of Parsons et al. [6] and most recently the one by Luque et al. [7]. Luque et al. proposed a parallel genetic algorithm in a ring topology where k subpopulations evolve independently. In this model a number of individuals is interchanged among these subpopulations with a pre-specified frequency.

On the other hand, one of the most important open problems in computational biology is the protein folding problem. The main goal in this problem is to predict the three-dimensional (3D) structure of proteins in their native states based

[1] base pairs

[2] http://www.phrap.org, March-2005

[3] http://www.tigr.org/tdb/tgi/software, March-2005

[4] http://www-cse.ucsd.edu/groups/bioinformatics, April-2005

solely in the linear sequence of amino-acids [8]. The research effort in this area has started in the 50's when Linus Pauling proposes the existence of a thermodynamically stable state with a helix structure for a particular class of proteins. This fact was confirmed experimentally in 1958 [1] for the protein known as myoglobin.

In the last decade several efforts to solve a simplified variant to this problem have been carried out. In this variant only hydrophobic interactions are taken into account, and the twenty amino-acids are classified into two groups: hydrophobic (H) and polar (P). In spite of these efforts, the problem has been solved only for short amino-acids sequences. There are many algorithms for the 2D model as well as for the 3D model. Some of these algorithms are exact [8, 10]. The algorithms are capable of finding optimal solutions in 3D cubic lattices for sequence of up to 88 amino-acids, with computation times ranging from minutes to hours. There exist also approximation algorithms [11-13]. These algorithms guarantee an approximation factor to the optimal solution. To date the best approximation ratio are as follows $1/3$ for 2D HP and $3/8$ for 3D HP. There also exist heuristics based algorithms. These algorithms can be divided into two categories: Evolutionary Algorithms [14-19], and Monte Carlo algorithms [20-24]. Although these algorithms do not guarantee the optimum they obtain good solutions efficiently.

Both problems, under their simplest models belong to the NP-hard class of problems [25], [32], [33], which basically implies that there is no known method to solve the problem in polynomial time in the number of fragments for the assembly problem, and in the number of residues for the folding problem.

The remainder of this chapter is organized as follows. Section 2 introduces the assembly problem and explains the genetic algorithm proposed to deal with this problem. Section 3 introduces the protein folding problem. Section 4 briefly describes the experiments and results achieved with the proposed algorithms. Finally, Section 5 states the conclusions and some ideas for future research.

2 The DNA Assembly Problem

Pevzner [26] proposes the following analogy to the problem of DNA fragment assembly: imagine we have many copies of a book; each of these copies is chopped in many, said ten million, pieces. Assume that 10% of these pieces are removed and the rest are splashed with ink. The target is to build a complete copy of the book. The pieces corresponds to the DNA fragments, the ink on some pieces corresponds to the base errors in the fragments, the missing pieces corresponds to the missing fragments. The goal of building a copy of the book corresponds to assembling the original DNA sequence.

A definition for the DNA assembly problem is as follows:

Definition 1. The input is a set of DNA fragments over the four letter alphabet $\sum = \{A,T,C,G\}$, the fragments can have different lengths from hundreds to thousands of bases. The output or solution is the shortest common superstring of this set $\sum$.

This definition does not consider that each of the input fragments is a result of a sequencing process which does not produce error free fragments. The base errors can be classified in substitutions and indels (insertions/deletions). Base substitution errors are in the range of 0.5 to 2% [4]. Furthermore, there is still another factor that makes the problem more complicated, namely the fragments orientation. With shotgun the fragments can come from any of the strands in the double-helix.

Taking into account all these factors we can redefine the problem as follows:

Definition 2. The input is a set P of reads over the alphabet $\sum = \{A,T,C,G\}$, each fragment p_i is of length d_i. The orientation of each p_i is unknown and each fragment can contain base errors. The output is a permutation of the p_i's of length $|P|$, such that the alignment score F(Π) among them is maximized.

$$F(\Pi) = \frac{\sum_{i=0}^{n-2} w(\Pi[i], \Pi[i+1])}{c(\Pi)} \tag{1}$$

where,

- d_i is the length of fragment i.
- Π is a permutation of the fragments.
- n is the total number of fragments.
- $\Pi[i]$ outputs the index of the fragment at position i within the permutation Π.

- $w(\Pi[i],\Pi[i+1])$ is a conditioned local alignment (alc) between fragment $\Pi[i]$ and the $\Pi[i+1]$.
- $c[\Pi]$ is the number of contigs given by permutation Π.

Equation 1 forces us to search for a small number of well-overlapped contigs.

We consider here a graph based model of the problem. We start by transforming the input of the problem into a complete directed graph G=(V,E) as follows:

- **Vertexes:** correspond to each of the k initial fragments that are denominated $p_1, p_2, ..., p_k$. Since the orientation for each of these fragments is unknown they cannot be assembled correctly unless all of them come from the same strand, which is very unlikely to happen. Due to this, it is proposed to add k segments that are the reverse complement of each fragment in the input to which we denominate $p_{1'}, p_{2'}, ..., p_{k'}$. Each of these $2k$ vertexes in V is assigned a weight of one.
- **Arcs:** Each arc $(p_i, p_j) \in$ E of the graph G has an assigned gain $g_{i,j}$ that represents the alc between the fragments p_i and p_j.
- **Objective:** To find a path r in the graph, such that the sum of visited arcs' gains is maximized, and the sum of the vertexes weights do not exceed k. This constraint is to guarantee that we are using no more than the k fragments originally in the input. For any path r a fragment p_i is considered to be in the path whenever the fragment $p_{i'}$ was not previously included in r, the same applies to $p_{i'}$.

Figure 1 shows a graph constructed from an input of three reads, the vertexes p_1, p_2 and p_3 represents to these fragments, the graph is completed with the vertexes: $p_{1'}$, $p_{2'}$ and $p_{3'}$ which represent the reverse complement of the three original fragments.

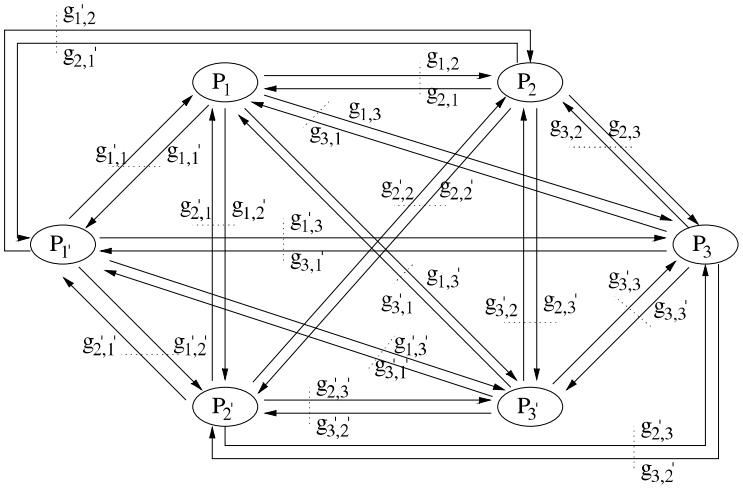

Fig. 2. *A graph constructed from an input of three DNA fragments (p_1, p_2, p_3) the reverse complement of these fragments are also considered ($p_{1'}, p_{2'}, p_{3'}$)*

2.1 The proposed Genetic Algorithm

The methodology proposed to solve the problem is based on a genetic algorithm, which constitutes one of the paradigms in evolutionary computation [27]. Next we show the pseudocode for the proposed algorithm (Algorithm 1).

Algorithm 1. GAssembler

Input: A set of DNA fragments
Output: A set of Contig(s) maximizing Equation 1

- **Preprocessing** { Generation of reverse complements; Overlaps Detection; Elimination of redundant fragments }

"GA starts"

- Generation of initial population
- Fitness Function Evaluation
- **Cycle starts**
- Selection
- Crossover
- Mutation
- Fitness Function Evaluation
- **After l iterations** { Fragments elimination Phase; and individuals' adjustment }
- **Cycle ends**
- Consensus

We present a brief explanation for the components of Algorithm 1.

Preprocessing. This preprocessing consist of three parts: the reverse complement computation, overlaps detection, and elimination of fragments. The first part is in charge of computing the reverse-complement sequence for each fragment in the input. The second part needs to compute the overlaps between fragments taking into account the substitution errors. Finally, the third part consist on eliminating all fragments that are subsegments, prefix, or suffix of any other fragment in the input.

Representation. An important issue to apply a GA in the resolution of a problem is to decide what representation to use. It is easy to see that a solution for the problem is a permutation of the fragments indexes, a permutation of $2k$ numbers. The codification we use is known as adjacency based representation [28].

Selection. We select the parents with the stochastic universal sampling algorithm ([27], page 62).

Crossover. A greedy crossover operator [29] is used. This operator produced high quality results for the sequencing by hybridization problem.

Mutation. The operator known as swap mutation for permutations is used ([27], page 45).

Fragment Elimination and adjustments of individuals. After l iterations we search for the longest contigs and analyze the fragments used on each of them. Once a fragment is used in a contig, if its reverse-complement appear anywhere downwards, then the latter is eliminated. After this operation is performed the individuals need to be adjusted by erasing their genes corresponding to reverse-complements that were already in contigs upwards.

The details for each component of the proposed algorithm can be found in [30].
In the next section we introduce the second problem to solve.

3 The Hydrophobic Polar (HP) model

An import nt assumption under this HP model is that the most significant force acting on a protein, and that contributes a great deal in the changes of the free energy, is the hydrophobic interaction [31]. This force is related to the property of certain molecules for repelling or attracting molecules of water.

When the amino-acids merge to form the peptide bond they free a water molecule. The resulting monomers are denominated *residues*, i.e. the residues are the amino-acids after having lost some atoms as a consequence of the binding. Depending on their side chain structure, these residues can be hydrophobic (repels water) or polar (attract water). Since many proteins, in their native states, are in an aqueous solution, the hydrophobic residues tend to concentrate in the interior part of the structure, defining in this way a particular folding.

Under this consideration the first simplification is undertaken. The 20 amino-acids are represented by only two monomers H or P, depending upon their affinity with water. The second simplification in this model is that each residue can occupy only the intersecting points in a 2D square lattice or a 3D cubic lattice. All self-avoiding walks on this lattice are allowed, i.e. the monomers are not allowed to collide with each other.

Under this model an H-H *contact* is defined as any pair of residues H's that are neighbors in the lattice but not in the sequence. To such contacts an energy of -1 is assigned. Since the natural state (native conformation) of a protein is given by is minimal energy free conformation, the optimal folding in the HP model is the one with the maximum number of H-H contacts (Figure 2).

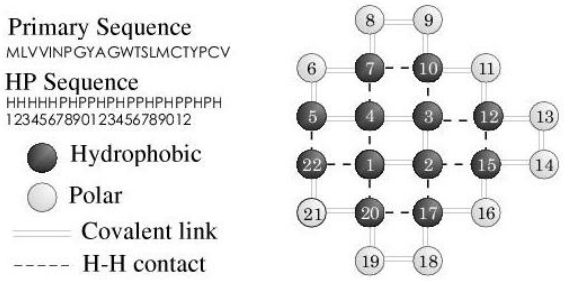

Fig. 3. *Optimal folding for the sequence MLVVINPGYAGWTSLMCTYPCV (HHHHHPHPPHPHPPHPHPPHPH), with a total of 12 H-H contacts, under the 2D HP model.*

3.1 Problem Definition

Under the HP model an amino-acid sequence of length n can be represented as $S = s_1 s_2 \ldots s_n$ where $s_i \in \{H, P\}, 1 \le i \le n$. The 3D structure of S is defined by a sequence of directions for the folding, starting in the first residue of the sequence. The codification to represent 3D conformations in the cubic lattice uses the symbols U, D, L, R, F, and B to denote the folding directions Up, Down, Left, Right, Front, and Back, respectively (see Figure 3).

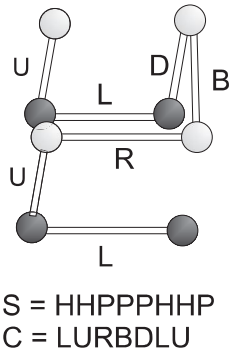

Fig. 4. *Optimal folding for the sequence S and its corresponding codification C, under the 3D HP model. The sequence starts with a hydrophobic residue located at the bottom right.*

A valid folding is the one where each location in the lattice has at most one residue, and each residue is connected to its neighbor(s) in the sequence, which is at the same time, a neighbor location(s) in the lattice.

Definition 3. Under the 3D HP model we can define the folding problem from an input/output point of view as follows:

Input: A sequence $S = s_1 s_2 s_3 \ldots s_n$ where $s_i \in \{H, P\}, 1 \le i \le n$, which represents a sequence of residues.

Output: A conformation or valid folding C on a 2D or 3D lattice that has the maximum number of H-H contacts.

In spite of all simplifications, it was proven that the problem in 2D HP is HP-hard [32], the same holds for the 3D HP model [33].

Yue and Dill have [8] proposed an algorithm to solve the problem. In their method they first find what is known as the optimal H-core. This core contains all hydrophobic residues in the sequence. Once they determine the volume and shape of this core, a combinatorial algorithm based on exhaustive search is applied to find the layout of the H residues inside the core. The latter problem of finding the layout of H residues given the optimal core is the second problem we are dealing with in this chapter.

Definition 4. Given the optimal H-core find the layout of hydrophobic residues inside the core, such that all amino-acids in the sequence remain connected.

3.2 The Proposed Algorithm

The algorithm has three main components: a global GA which evolves the layouts inside the core, a Monte Carlo method which helps to avoid exhaustive search of all possible combinations inside the core, and finally, the internal GA that helps to connect polar segments (outside the core) between pairs of hydrophobic residues (inside the core).

Algorithm 2. P-Folder

1. define the frame and the H-core
2. repeat n times
3. initialize the population
4. Start global GA: Initialize the Population
5. Fitness Evaluation
6. search for paths connecting H-H residues
7. run internal GA to compute the total number of connections
8. select the parents
9. apply MC based crossover (MC to handle constraints)
10. Fitness Evaluation
11. search for paths connecting H-H residues
12. run internal GA to compute the total number of connections
13. apply MC based mutation (MC to handle constraints)
14. end global GA
15. apply local optimization
16. end repeat
17. start refinement phase
18. select the best $\alpha\%$ individuals from the previous phase
19. compute the longest connected segment
20. repeat global GA with fixed segments
21. end refinement phase
22. output the best individual

The details for the proposed algorithm to deal with the 3D HP protein folding problem can be found in [34].
In Line 1 we need to define the frame and the H-core. This is achieved by the method proposed by Yue and Dill [8].

Generation of Initial Population. Once the H-core is defined a random layout of the H residues inside the core is performed for each individual. Note that it is not mandatory that all residues are connected at this stage.

Fitness Computation. The fitness is given by the total number of connected residues. Since the number of possible paths between pairs of residues grows exponentially with the longest length of a P segment, an internal GA is used to compute the connections. That is, the objective function is computing by running an internal genetic algorithm.

4 Computational Experiments and Results

4.1 Results for the DNA assembly problem

Engle and Burks [35] proposed a set of tools denominated *genfrag* to generate instances to test new assemblers. Here we use an implementation of this generator. We deal with sequences of up to 20000 base pairs.

The results were evaluated considering the number of sequences generated by the algorithm and the similarity of these sequences with the original one. On instances where the error was at the level of 5% the genetic algortihm clearly outperforms the TIGR [30].

4.2 Results for the protein folding problem

The results for the protein folding problems are also encouraging, out of 13 sequences 12 were laid out in their optimal positions. Among these; 10 sequences are of 48 residues, one of 64, one of 67, and one of 88 residues. The instance with 88 residues were the only one on which the algorithm could not find the optimal layout [34].

5 Conclusions

New strategies based on genetic algorithms and heuristics to tackle the DNA assembly and a simplified model of the protein folding problems have been proposed. The use of problem specific knowledge inside the algorithm has shown to be very effective for solving complex problems. The protein folding problem, under the 3D cubic lattice model, remains a computationally challenging problem

Future work will be focused on applying these methods to solve bigger instances for both problems. Another important open Bioinformatics problem to explore with these ideas has to do with: given a 3D structure of a protein, predict its function.

Acknowledgments

This research was partially supported by CONACyT under grant C01-45811 and by the Laboratotio Franco-Mexicano de Informática.

References

1. Mihai Pop, L. Salzberg, and M. Shumway. Genome sequence assembly: algorithms and issues. *IEEE computer*, 35(7):47--54, 2002.
2. F. Sanger and A. R. Coulson. The use of thin acrylamide gels for dna sequencing. *FEBBS*, 87:107--110, 1978.
3. Xiaoqiu Huang and Anup Madan. Cap3: a dna sequence assembly program. *Genome Research*, 9:868--877, 1999.
4. Ken Stanley Jonathan Butler Sante Gnerre Evan Mauceli Bonnie Berger Jill P. Mesirov Serafin Batzoglou, David B. Jaffe and Eric S. Lander. Arachne: A whole-genome shotgun assembler. *Genome Research*, (12):177--189, 2002.
5. Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. An eulerian path approach to dna fragment assembly. *PNAS*, 98(17):9748--9753, 2001.
6. Rebecca J. Parsons, Stephanie Forrest, and Christian Burks. Genetic algorithms, operators, and dna fragment assembly. *Machine learning*, 21:11--33, 1995.
7. Gabriel Luque, Enrique Alba, and Sami Khuri. *Parallel algorithms for solving the fragment assembly problem in DNA strands*, chapter Parallel computing for bioinformatics and computational biology. John Wiley & Sons. New York, pages 287--304. En: A. Y. Zomaya (ed.), 2005.
8. K. Yue and K. Dill. Sequence-structure relationships in proteins and copolymers. *Phys. Rev. E*, 48(3):2268--2278, Septiembre 1993.
9. [1] J. Kendrew, G. Bodo, H. Dintzis, R. Parrish, H. Wyckoff, and D. Phillps. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662--666, Mar 1958.
10. K. Yue and K. Dill. Forces of tertiary structural organization in globular proteins. *Proc Natl Acad Sci*, 92(1):146--150, Enero 1995.
11. W. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within thee-eights of optimal. In *Proceedings of the twenty-seventh annual ACM symposium on theory of computing*, pages 157--168, 29 de mayo al 1 de junio, Las Vegas, Nevada, Junio 1995.
12. A. Newman. A new algorithm for protein folding in the hp model. In *Proceedings of the thirteenth annual ACM-SIAM symposium on discrete algorithms*, pages 876--884, 6 al 8 de enero, San Francisco, CA, Enero 2002.
13. A. Newman and M. Ruhl. Combinatorial problems on strings with applications to protein folding. In *Proceedings of the sixth Latin American symposium on theoretical informatics*, pages 369--378, 5 al 8 de abril, Buenos Aires, Argentina, Abril 2004.
14. T. Jiang, Q. Cui, G. Shi, and S. Ma. Protein folding simulations of the hydrophobic-hydrophilic model by combining tabu search with genetic algorithms. *Journal of Chemical Physics*, 119(8):4592--95, 2003.
15. R. Konig and T. Dandekar. Improving genetic algorithms for protein folding simulations by systematic crossover. *BioSystems*, 50(1):17--25, 1999.
16. N. Krasnogor, W. Hart, J. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In *Proceedings of the genetic and evolutionary computation conference*, pages 1596--1601, 13 al 17 de julio, Orlando, Florida, Julio 1999.
17. A. Patton, W. Punch, and E. Goodman. A standard ga approach to native protein conformation prediction. In *Proceedings of the sixth international conference on genetic algorithms*, pages 574--581, julio, San Francisco, CA, Julio 1995.

18. A. Shmygelska and H. Hoos. An improved ant colony hoptimisation algorithm for the 2d hp protein folding problem. In *Proceedings of te sixteenth conference of the Canadian society for computational studies of intelligence*, pages 400--417, junio, Halifax, Canada, Junio 2003.
19. R. Unger and J. Moult. A genetic algorithm for 3d protein folding simulations. In *Proceedings of the fifth international conference on genetic algorithms*, pages 581--588, 17 al 22 de julio, Illinois, Julio 1993.
20. T. Beutler and K. Dill. A fast conformational search strategy for finding low energy structures of model proteins. *Protein Sci.*, 5:2037--2043, 1996.
21. U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler. Testing a new monte carlo algorithm for protein folding. *Proteins*, 32(1):52--66, 1998.
22. H. Hsu, V. Mehra, W. Nadler, and P. Grassberger. Growth algorithms for lattice heteropolymers at low temperatures. *Journal of Chemical Physics*, 118(1):444--451, Enero 2003.
23. F. Liang and W. Wong. Evolutionary monte carlo for protein folding simulations. *Journal of Chemical Physics*, 115(7):3374--3380, 2001.
24. R. Ramakrishnan, B. Ramachandran, and J. Penky. A dynamic monte carlo algorithm for exploration of dense conformational spaces in heteropolymers. *Journal of Chemical Physics*, 106(6):2418--2424, 1997.
25. Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, Nueva York, NY, 1997.
26. P. A. Pevzner. *Computational molecular biology: An algorithmic approach*. The MIT Press, London, 2000.
27. A.E. Eiben and J.E. Smith. *Introduction to evolutionary computing*. Springer, Alemania, 2003.
28. John J. Grefenstette, Rajeev Gopal, Brian J. Rosmaita, and Dirk Van Gucht. Genetic algorithms for the traveling salesman problem. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 1:160--168. Lawrence Erlbaum Associates, 1985.
29. C. Brizuela, L. González, and H. Romero. An improved genetic algorithm for the sequencing by hybridization problem. In *Proceedings of the 3rd European Workshop on Evolutionary Computation in Bioinformatics, EvoBio, LNCS*, volume 3005, pages 11--20. Springer-Verlag, 2004.
30. Milton Rodriguez-Zambrano. *Un algoritmo genetico para el ensamble de secuencias de ADN*. CICESE, Ensenada, 2005.
31. K. Dill. Dominant forces in protein folding. *Biochemistry*, 29(31):7133--7155, Agosto 1990.
32. Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423--466, 1998.
33. B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. In *Proceedings of the second annual international conference on computational molecular biology*, pages 30--39, 22 al 25 de marzo, New York, Marzo 1998.
34. Jorge E. Luna-Taylor. *Un algoritmo evolutivo hibrido para el problema de plegamiento de proteinas bajo el modelo hidrofobico polar en tres dimensiones*. CICESE, Ensenada, 2006.
35. Michael L. Engle and Christian Burks. Artificially generated data sets for testing dna sequence assembly algorithms. *Genomics*, 16:286--288, 1993.