Evolving L-Systems to Capture Protein Structure Native Conformations

Gabi Escuela[1], Gabriela Ochoa[2] and Natalio Krasnogor[3]

[1,2] Department of Computer Science, Simon Bolivar University, Caracas, Venezuela
gabiescuela@netuno.net.ve, gabro@ldc.usb.ve
[3] School of Computer Science and I.T., University of Nottingham
Natalio.Krasnogor@nottingham.ac.uk

Abstract. A protein is a linear chain of amino acids that folds into a unique functional structure, called its native state. In this state, proteins show repeated substructures like alpha helices and beta sheets. This suggests that native structures may be captured by the formalism known as Lindenmayer systems (L-systems). In this paper an evolutionary approach is used as the inference procedure for folded structures on simple lattice models. The algorithm searches the space of L-systems which are then executed to obtain the phenotype, thus our approach is close to Grammatical Evolution. The problem is to find a set of rewriting rules that represents a target native structure on the lattice model. The proposed approach has produced promising results for short sequences. Thus the foundations are set for a novel encoding based on L-systems for evolutionary approaches to both the Protein Structure Prediction and Inverse Folding Problems.

1 Introduction

The Protein Structure Prediction Problem (PSP) is among the most outstanding open problems in Biochemistry. A successful approach for efficient and accurate prediction would hasten a new era for biotechnology. A protein is as a linear sequence of units, called amino acids, that under certain physical conditions folds into a unique functional structure known as the native state or tertiary structure. This native state is the key to understanding a proteins' functionality in a living organism as an enzyme, a storage, transport, messenger, antibody, or regulation molecule. The simplest models for studying the properties of protein folding and structure prediction are based on lattices (of 2 or 3 dimensions), these models capture the essential aspects of the folding process while keeping low computational costs. The on-lattice hydrophobic-hydrophilic (HP) model, assumes the hydrophobic effect of amino acids as the main force governing folding.

The correspondence between amino acids and positions within a lattice is called *embedding* of the protein. It was shown that finding the embedding of a protein is NP-hard even for very simple lattice models [7,33]. Thus, the use of heuristics and approximation algorithms became the most promising approach for the PSP. In particular, several evolutionary algorithms have been suggested for solving this problem [12,18,19,27,34]. All these approaches employ a direct encoding of the folded chain (See Section 2).

M. Keijzer et al. (Eds.): EuroGP 2005, LNCS 3447, pp X-XY, 2005, pp. 73 – 83, 2005.

This paper suggests a *novel encoding* scheme for the PSP based on Lindenmayer systems. The rationale for this is twofold:

1. A protein structure often exhibits a high degree of regularity, with a wealth of secondary structures, preferred motifs, and tertiary symmetries [8]; also, proteins have been compared to fractals [32]. This is consistent with the recursive nature of L-systems where rewriting rules lead to modular, auto-similar structures.
2. It is not clear that the encoding currently used in evolutionary algorithms for HP models, namely, internal coordinates (see section 2.1) are suitable for crossover and building block transfer between individuals [4,15,16].

An evolutionary algorithm is proposed as the inference procedure for folded structures under the HP model in 2D lattices. *The problem is to find a set of rewriting rules (an L-system) that captures a target folded structure (which represents the native state for a given protein) on the selected lattice model*.

Evolutionary algorithms have been successfully applied to a variety of design problems, but it is not clear whether evolutionary techniques can scale to the complexities of real world designs. It has been argued that a generative or grammatical encoding scheme, (i.e. an encoding that specifies how to construct the phenotype, instead of a direct encoding of the phenotype) can achieve greater scalability through self-similar and hierarchical structure [1,2,10]. Moreover, by reusing parts of the genotype while generating the phenotype, a generative encoding is a more compact encoding of a solution. These approaches to encoding have had an enormous success; we point the reader to [25,30,31] for a general overview of grammatical evolution and to [26] for an application of grammatical evolution to a problem related to the one we focus on here.

L-systems as a generative encoding have been used in previous applications of evolutionary algorithms to problems in biology, medicine, engineering, and computer graphics. The production of plant structures [3,6,13,24,28] has been the most studied case; where results have shown the usefulness of this encoding, both to obtain structures resembling natural organisms, and in the generation of artificial designs with novel features. Furthermore, L-systems grammars have proved to be a powerful genotype encoding to represent blood circulation of the human retina [14], physical design of tables, robots, and virtual creatures [9,11], and in the design of transmission towers [29].

We proceed as follows: Section 2 provides the theoretical basis of the PSP; section 3 describes the mathematical formalism of L-systems. The proposed approach is presented in Section 4. Section 5 describes the experiments and results; and finally section 6 concludes and comments on future work.

2 The Protein Structure Prediction Problem Simplified

Proteins are the building blocks and functional units of all biological systems. There are 20 naturally occurring amino acids that make up protein chains. The amino acid's chain of a protein is known as its primary structure and usually contains about 30 to 400 acids. The primary structure folds in space and forms secondary structures. These secondary structures present specific signatures like α-helices and β-sheets. In turn,

secondary motifs fold yet again and aggregate in space giving raise to a 3D conformation, the tertiary structure. The tertiary structure conforms a very specific geometric pattern (the native state).

2.1 The HP Model

In the HP model [5], only two types of monomers are distinguished: *hydrophobic* (H), and *polar* or *hydrophilic* (P). The hydrophobic monomers tend to occupy the center of the protein, staying close to each other to avoid surrounding water, whereas the polar residues are attracted to water and are frequently found on the convex hull of the native state. The set of valid protein structure conformations is the space of all self-avoiding paths (on a selected lattice, e.g., square 2D, triangular, cubic, diamond, etc.), with each amino acid located on a lattice bead. Hydrophobic units that are adjacent in the lattice but non-adjacent in the sequence (also called non-local H-H contacts) add a constant negative factor (generally ε=-1) and all other interactions are ignored. The native state is thought to be the global energy minimum.

In the HP model, the structures can be represented by Cartesian coordinates, internal coordinates or distance geometry. We concentrate here on internal coordinates, which can be defined as absolute or relative. Under the absolute encoding, the structures are represented by a list of absolute moves. In a 2D square lattice, for example, a structure *s* is encoded as a string $s = \{$**U**p, **D**own, **L**eft, **R**ight$\}^+$. When using a relative coordinates, each move is interpreted in terms of the previous one, like in LOGO turtle graphics; a structure *s* is encoded as a string $s = \{$**F**orward, Turn**L**eft, Turn-**R**ight$\}^+$. Designing with black the hydrophobic residues and white the polar ones, the structure of Figure 1 is coded either as *s =RDDLULDLDLUURULURRD* (absolute encoding) or s = *RFRRLLRLRRFRLLRRFR* (relative encoding), with 9 non-local H-H contacts.

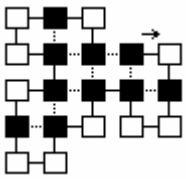

Fig. 1. Native structure in the square 2D lattice for the primary sequence HPHPPHHPHPPHPHHPPHPH. The arrow indicates the starting point, and the dotted lines the non-local H-H contacts

3 L-Systems

Aristid Lindenmayer (a biologist) proposed in 1968 an axiomatic foundation for biological development called L-systems [21]. More recently, L-systems have found several applications in computer graphics [28]; two principal areas include generation of fractals and realistic modeling of plants. Central to L-systems, is the notion of rewriting, where the idea is to define complex objects by successively replacing parts of a simple object using a set of rewriting rules or productions. The rewriting can be carried out recursively. The most extensively studied and best understood rewriting

systems operate on character strings. The essential difference between the most known Chomsky grammars and L-systems lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel, replacing simultaneously all letters in a given word. This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time.

3.1 D0L-Systems

The simplest class of L-systems is the *D0L-systems* (deterministic and context free). To provide an intuitive understanding of the main idea, let us consider the example given by Prusinkiewicz and Lindenmayer [28] (See Figure 2.).

> "Lets us consider strings built of two letters a and b (they may occur many times in a string). For each letter we specify a rewriting rule. The rule $a \rightarrow ab$ means that the letter *a* is to be replaced by the string *ab*, and the rule $b \rightarrow a$ means that the letter *b* is to be replaced by *a*. The rewriting process starts from a distinguished string called the axiom. Let us assume that it consist of a single letter *b*. In the first derivation step (the first step of rewriting) the axiom *b* is replaced by *a* using production $b \rightarrow a$. In the second step *a* is replaced by *ab* using the production $a \rightarrow ab$. The word *ab* consist of two letters, both of which are simultaneously replaced in the next derivation step. Thus, *a* is replaced by *ab* , *b* is replaced by *a*, and the string *aba* results. In a similar way (by the simultaneous replacement of all letters), the string *aba* yields abaab which in turn yields *abaababa*, then *abaababaabaab*, and so on."

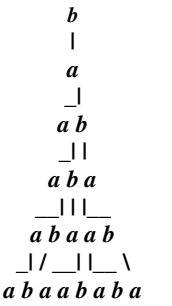

Fig. 2. A D0L-system derivation example

4 Method

Our proposed approach uses an evolutionary algorithm that, given a target structure in internal relative coordinates *(input)***,** will evolve an L-system L *(output)* that, once evaluated, would produce a string that matches the original target. For instance, the end-product of the EA run for the structure in Figure 1 would be an L whose termination word is *RFRRLLRLRRFRLLRRFR*.

A generational EA with linear ranking selection and elitism was used to evolve sets of rewriting rules or L-systems that capture a target structure. As the variation opera-

tors, a recombination and three mutation operators were implemented. Two stopping criteria were considered: (i) if an individual arises the maximum fitness, that is, its L-system grammar exactly represents the target folding; and (ii) a predefined maximum number of generations is reached. The genotype encoding, initial population, genetic operators, and fitness evaluation are described below. Furthermore, the specific values for the various algorithm's parameters used in the experiments, are listed in Section 5.

4.1 Genotype Encoding and Initial Population

The L-system's alphabet will depend on the lattice and coordinate system used. For the experiments reported here, we selected the square 2D lattice with relative coordinates. Thus, the terminal characters are the symbols {F, L, R}.

Genotypes are encoded using D0L-systems with the following characteristics:

$$\text{Alphabet: } \Sigma=\Sigma_t \cup \Sigma_{nt} \quad \text{where } \Sigma_t=\{F,L,R\} \text{ terminal characters and } \Sigma_{nt}=\{0,1,2,...,m\text{-}1\} \text{ non-terminal characters representing rewriting rules}$$

$$\text{Axiom: } \alpha = S \qquad S \in \Sigma^+$$

$$\text{Rewriting rules: } \quad W_{0,1,2,...,m-1}: w, \text{ where } w \in \Sigma^+$$

A string representing the axiom, the number of rewriting rules and the strings representing each rule, determine the genotype of an individual. The maximum lengths of the axiom and rules, as well as the number of rules are parameters that will depend on the length of the original folding. As the maximum values are held as parameters, the specific values for each individual within a population may differ.

Let *max_r*, *max_la*, and *max_lr* be the maximum number of rules, and maximum string lengths for the axiom and production rules respectively; an individual of the initial population is generated as follows: the number of rules is randomly selected in the range 1 to *max_r*, this define the non-terminal characters allowed for the individual. The axiom is a randomly generated string of symbols of maximum length *max_la* where each symbol is selected with uniform distribution from the alphabet Σ. Thereafter, each rule is generated in a similar way as the axiom, with a maximum length of *max_lr*.

4.2 Genetic Operators

Recombination takes two individuals, *p1* and *p2* as parents and creates one offspring, *o*, by copying the axiom of *p1* and selecting rules from either *p1* or *p2* with a probability of 0.5; this recombination operator resembles *uniform* crossover, where the interchanged genes are complete rules. To maintain consistency, if a selected rule to conform *o* makes reference to a symbol (rule) not defined in *o*, then a repair operator changes that symbol for a suitable symbol (either terminal or non-terminal). Fig. 3 shows an example of how this operator is applied.

A mate selection strategy (*dissasortative* mating) was also implemented as a mechanism for increasing the population genetic diversity. Dissasortative mating was implemented as follows: when selecting two individuals for a crossover, the first par

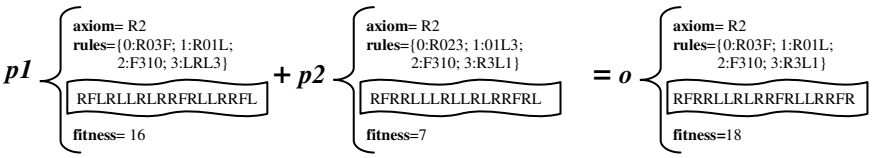

Fig. 3. Genotype, phenotype and fitness from parents and offspring in a recombination where *o* inherits the rules 0,1 from *p1* and 2,3 from *p2*

ent was selected as usual. To chose the second parent, a set of *s* (*scan* size) individuals were selected using the GA fitness-based selection method. Thereafter, the similarity between each of these *s* phenotypes and the first parent was computed, the phenotype with less similarity was chosen. For the experiments reported here, Hamming distance was used as the similarity measure, and the scan size *s* was set to 5.

Three mutation operators were implemented that perform: (i) addition, (ii) deletion, or (iii) modification of a single symbol that conforms either the axiom or the rewriting rules of each individual. When a mutation is to be performed, 60 % of times it will be a modification, 30 % an addition, and 10 % deletion.

4.3 Derivation Process, Post-processing and Fitness Calculation

For computing an individual's fitness, its L-system is derived. That is, starting from the axiom, the rewriting rules are applied in a parallel and iterated way, until either the number of terminal characters becomes equal to or greater than the string length of the target folding; or no more non-terminal characters are present in the string. Thereafter, a post-processing phase prunes the non-terminal symbols in the string to produce the phenotype. The fitness value will be the number of matches between the produced phenotype and the target folding, that is a generalized Hamming distance. So, the minimum fitness is 0 and the maximum is the length of the desired folding.

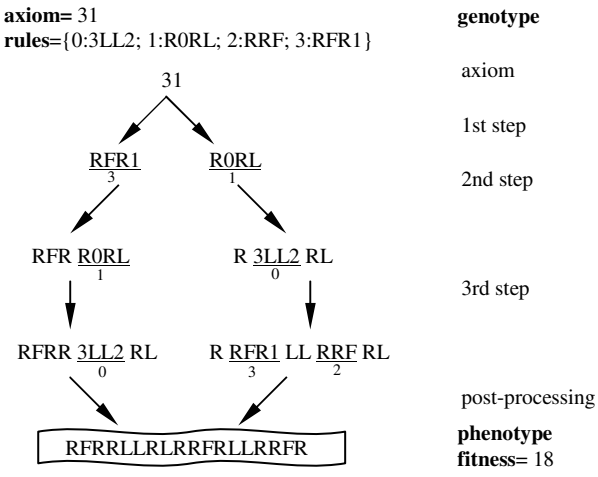

Fig. 4. Example of a derivation process

Figure 4 illustrates the derivation process for an individual (Solution 1 of Table 3) Three derivation steps, and the final result after a post-processing stage, are shown. In this case, the phenotype matches exactly the target *RFRRLLRLRRFRLLRRFR*.

5 Experiments and Results

We selected four protein instances from the HP benchmark available at http://www.cs.nott.ac.uk/~nxk/hppdb.html. Thereafter, their foldings embedded in the 2D square lattice with relative coordinates, were found using MAFRA (Memetic Algorithm FRAmework) [17]. Each of the obtained foldings was set as the target for our evolutionary approach, using the parameters listed in Table 1.

Table 1. Parameter values used for the experiments

Parameter	Value
Max. Number of Generations	2000
Population Size	50
Mating Strategy	Disassortative 5
Mutation rate (per symbol) Axiom	0.05
Mutation rate (per symbol) Rules	0.05
Recombination rate	1.00
Max. Number of Rules	4-5
Max. Length for Axiom	3
Max. Length for Rules	5

Table 2 summarises the results obtained for the selected four instances. The number of successes (runs that produced the target folding exactly) out of 50 runs, and a selected solution (L-system) are shown for each instance.

Table 2. Results for 4 instances (50 runs each)

Instance	Length	Successes	One Solution
HPHPPHHPHPPHPHHPPHPH→ RFRRLLRLRRFRLLRRFR	18	5/50 (4 rules)	See Table 3
HHHPPHPHPHPPHPHPHPPH → RRFRFRLFRRFLRLRFRR	18	3/50 (4 rules)	axiom = R2 4 rules = {0:RLR; 1:3F32L; 2:1FR33; 3:R102}
HHPPHPPHPPHPPHPPHPPHPHH → RLLFLFFRRFLLFRRLRFFRRF	22	0/50 (4 rules) 1/50 (5 rules)	axiom = 1R 5 rules = { 0:4LF3; 1:RL243; 2:00F3; 3:RRFL; 4:0R14F}
PPHPPHHPPPPHHPPPPHHPPPPHH → FFRRFFFLLFFFFRRFFFFLLFF	23	1/50 (5 rules)	axiom= 32 4 rules = {0:20R2; 1:132F; 2:FF012; 3:0FLL}

Table 3. Some results obtained for the folding *RFRRLLRLRRFRLLRRFR*

Solution	Axiom	Rewriting rules			
1	31	0:3LL2	1:R0RL	2:RRF	3:RFR1
2	31	0:3L23	1:R0L1	2:1LR	3:RFR1
3	31	0:3LLR	1:R02L	2:23	3:RFR1
4	021	0:1R2LR	1:R1F1R	2:1LLR1	
5	11	0:2210L	1:RF30R	2:LR2	3:RRL
6 (bs)	01F	0:RFR1	1:2L2	2:R0L	
7	RF3	0:3RFR	1:312L (nu)	2:RRLLR	3:20L0R
8	RF3	0:R3L0	1:0L2R1	2:231RF	3:0R20L
9	RF0	0:R1LL0	1:0R2FR	2:LRR	
10	RF2	0:12RR0	1:RLL3R	2:R1F0	3:RL12R
11	12	0:RL10	1:RF2R	2:30L3L	3:12R1
12	30	0:RFR10	1:LL3R	2:3F13 (nu)	3:0R1LR
13	30	0:R32	1:01L2	2:030R	3:RFR1L

(bs: best solution, since it has fewer and shorter rules)
(nu: not used)

Table 3 shows results for the first target folding (length 18). Several L-systems (of 3 and 4 rules) that successfully capture the folded structure were found by the evolutionary algorithm. Some solutions (7 and 12) evolved rules that were not used in the derivation process. We distinguished solution 6 as the best obtained in this set, since it has fewer and shorter rules. Notice that some substrings that appear several times in the folded chain (e.g. ***RFR***) also are present as part of the evolved rules. This supports the idea that the L-system captures the natural occurring substructures in the protein.

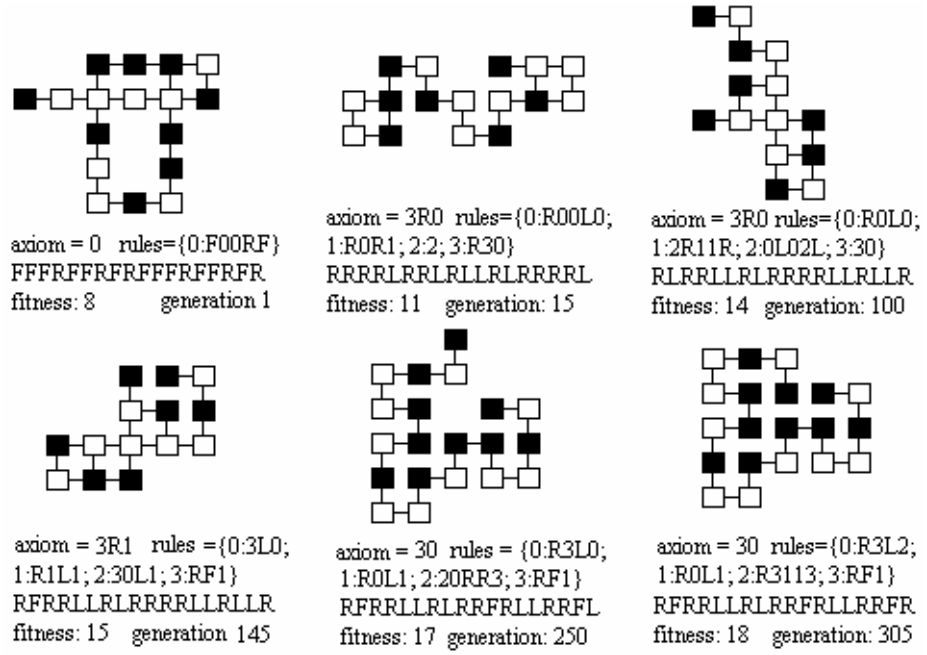

Fig. 5. Evolutionary progression towards the target structure (1st instance in Table 2)

Figure 5 shows the progression towards the target structure (1st instance in Table 2) as generations go by. The axiom, rules, fitness value, internal coordinates word, and graphical representation are displayed.

We would like to note that during the EA run, the production of illegal (not self-avoiding) structures was allowed (see for example the structure in generation 1 and 100 in Fig.5). However a successful L-system is only accepted when it is fully self-avoiding (like in generation 305). Also note that a given target structure may have various internal coordinates' representations (modulus rigid rotations), and that various distinct L-systems could produce the same internal coordinates word.

It is worth mentioning that the level of difficulty for evolving an adequate L-system widely varies with the instance selected. Additional to the folding's length; some instances seem more difficult than others. Our intuition is that the level of modularity and repetition within the protein folding varies across the space of possible structures.

6 Discussion

An evolutionary algorithm discovered L-systems that capture a target folding under the HP model in 2D lattices. These promising results set the foundations of a novel generative encoding for evolutionary approaches to both the protein structure prediction problem and inverse protein folding problem. We suggest that a generative encoding (i.e. a developmental approach for producing structures using a set of grammatical rewriting rules – L-system) may have better scaling properties than the direct internal coordinates encoding [1,2,10]. As noted in the previous section there are several symmetries that could be explicitly handled as to enhance the evolutionary search. Further work should test this hypothesis. Longer chains and 3D lattices should also be explored. The final goal will be to use an evolutionary approach with an L-system's encoding to solve challenging instances of the protein structure prediction and to evolve primary sequences which fold to specific native states (inverse folding).

References

1. Bentley, P. J.: Exploring component-based representations: the secret of creativity by evolution? *Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000)*, 2000.
2. Bentley, P. J. and S. Kumar.: Three ways to grow designs: A comparison of embryogenies of an evolutionary design problem. In Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiel, and Smith, editors, *Genetic and Evolutionary Computation Conference*, pages 35–43, 1999.
3. Curry, R.: On the Evolution of Parametric L-systems. Technical Report 1999-644-07. University of Calgary, Canadá, 1999.
4. De la Canal, E., Krasnogor, N., Marcos, D., Pelta, D. and Risi, W.: Encoding and Cross over Mismatch in a Molecular Design Problem. *Proceedings of Artificial Intelligence in Design '98 (AID98).* 1998.
5. Dill, K.: Theory for the folding and stability of globular proteins. *Biochemistry*, 24:1501, 1985.
6. Ebner, M., Grigore, A., Heffer, A., y Albert, J.: Coevolution produces an arms race among virtual plants. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi (editors): *Proceedings of the Fifth European Conference on Genetic Programming (EuroGP 2002)*, Kinsale, Ireland, pp. 316-325, Springer-Verlag, 2002.

7. Fraenkel, A.: Complexity of protein folding. *Bull. Math Biol*, 55:1199-1210, 1993.
8. Helling, R., Li, H., Miller, J., Mélin, R., Wingreen, N., Zeng, C., and Tang, C.: The Designability of Protein Structures. *J. Mol. Graph. Model.* 19, 157. 2001.
9. Hornby, G. and Pollack, J.: Evolving L-Systems to Generate Virtual Creatures. *Computers and Graphics*. 25:6, pp 1041-1048, 2001.
10. Hornby, G. and Pollack, J.: The advantages of Generative Grammatical Encodings for Physical Design. *Congress on Evolutionary Computation 2001 (CEC01)*, 2001.
11. Hornby, G., Lipson, H., and Pollack, J.: Evolution of Generative Design Systems for Modular Physical Robots. *IEEE International Conference on Robotics and Automation (ICRA)*, 2001.
12. Khimasia, M. and Coveney, P.: Protein structure prediction as a hard optimization problem: The genetic algorithm approach. In *Molecular Simulation*, volume 19, pages 205-226, 1997.
13. Kókai, G. Tóth, Z. and Ványi, R.: Evolving Artificial Trees described by Parametric L-Systems. In *Proc. IEEE Canadian Conference on Electrical & Computer Engineering*, Shaw Conference Centre, Canada, pages 1722-1728, 1999.
14. Kókai, G., Tóth, Z. and Ványi, R.: Modelling Blood Vesels of the Eye with Parametric L-Systems using Evolutionary Algorithms. In the *Proc Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*, Denmark published by Springer-Verlag LNCS series 1620 pages 433-443, 1999.
15. Krasnogor, N. Pelta, D., Martinez, P. and De la Canal, E.: Genetic Algorithms for the Protein Folding Problem: a Critical View. *Engineering of Intelligent Systems (E.I.S. 98)*, Proceedings of the conference, 1997.
16. Krasnogor, N., Marcos, D., Pelta, D. and Risi, W.: Protein Structure Prediction as a Complex Adaptive System. *Proceedings of Frontiers in Evolutionary Algorithms (FEA98)*, 1998.
17. Krasnogor, N. and Smith, J.: MAFRA: A Java Memetic Algorithms Framework. In A. Wu, editor, Workshop Program, Proceedings of the 2000 Genetic and Evolutionary Computation Conference. Morgan Kaufmann, 2000.
18. Krasnogor, N., Blackburnem, B., Hirst, J. and Burke, E.: Multimeme Algorithms for Protein Structure Prediction. *Proceedings of Parallel Problem Solving From Nature. Lecture Notes in Computer Science*, 2002.
19. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms. Ph.D. Dissertation, University of the West of England, 2002. Available [On-line] on http://www.cs.nott.ac.uk/~nxk/papers.html
20. Liang, F. and Wong, W.: Evolutionary Monte Carlo for protein folding simulations. *Journal of Chemical Physics*, 115(7):3374-3380, 2001.
21. Lindenmayer, A.: Mathematical models for cellular interactions in development, parts I-II. *Journal of Theoretical Biology* 18: 280-315, 1968.
22. Mock, K.: Wildwood: The Evolution of L-Systems Plants for Virtual Environments. In Proc ICEC 98. *IEEE-Press*, Anchorage, Alaska, 1998.
23. Noser, H., Stucki, P., Walser, H.: Integration of Optimization by Genetic Algorithms into an L-System Animation System. *Proceedings of Computer Animation* 2001, Seoul, Korea, November 7-8, 2001, pp. 106-112, 2001.
24. Ochoa, G.: On genetic algorithms and Lindenmayer Systems. In A. Eiben, T. Baeck, M. Schoenauer, y H.P. Schwefel, editors. *Parallel Problem Solving from Nature V*, pages 335-344. Springer-Verlag, 1998.
25. O'Neil M and Ryan C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Series : Genetic Programming , Vol. 4. Springer, 2003.
26. Ortega A., Dalhoun A. and Alfonseca M.: Grammatical Evolution to Design Fractal Curves with a Given Dimension. IBM Journal Res & Dev, Vol7, Nro 47, 2003.

27. Patton, A., Punch, W. and Goodman, E.: A Standard GA approach to native protein conformation prediction. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 574-581. Morgan Kauffman, 1995.
28. Prusinkiewicz, P. and Lindenmayer, A.: *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.
29. Rudolph, S. and Alber, R.: An Evolutionary approach to the inverse problem in Rule-based design representations. *Proceedings of the 7th International Conference on Artificial Intelligence in Design (AID'02)*, Cambridge University, UK, 2002, Kluwer Academic Publishers.
30. Ryan C., Collins J.J. and O'Neil M.: Grammatical Evolution: Evolving Programs for an Arbitrary Language. Proceedings of the First European Workshop on Genetic Programming. 1998.
31. Ryan C., O'Neil M. and Collins J.J.: "Grammatical Evolution: Solving Trigonometric Identities". Proceedings of Mendel 1998: 4th International Mendel Conference on Genetic Algorithms, Optimisation Problems, Fuzzy Logic, Neural Networks, Rough Sets. 1998
32. Sadana, A. and Vo-Dinh, T.: Biomedical implications of protein folding and misfolding. *Biotechnol. Appl. Biochem.* 33, (7–16). Great Britain, 2001.
33. Unger, I. and Moult, J.: Finding the lowest free energy conformation of a protein is an NP-hard problem: Proof and implications. *Bull Math. Biol*., 55:1183-1198, 1993.
34. Unger, I. and Moult, J.: Genetic Algorithms for protein folding simulations. *Journal of Molecular Biology*, 231 (1);75-81, 1993.