



**Repositorio Institucional de la Universidad Autónoma de Madrid**

<https://repositorio.uam.es>

Esta es la **versión de autor** de la comunicación de congreso publicada en:  
This is an **author produced version** of a paper published in:

IBM Journal of Research and Development 47.4 (2003): 483 – 493

**DOI:** <http://dx.doi.org/10.1147/rd.474.0483>

**Copyright:** © 2003 IBM

El acceso a la versión del editor puede requerir la suscripción del recurso  
Access to the published version may require subscription

# Grammatical Evolution to Design Fractal Curves with a given Dimension

Alfonso Ortega, Abdellatif Abu Dalhoum, Manuel Alfonseca

Universidad Autónoma de Madrid,

Campus de Cantoblanco,

28049 Madrid, Spain

{ Alfonso.Ortega, Abdel.Latif, Manuel.Alfonseca@ii.uam.es }

## **Abstract**

*Lindenmayer Grammars have been applied frequently to represent fractal curves. In this work, the ideas behind Grammar Evolution are used to automatically generate and evolve Lindenmayer Grammars that represent fractal curves with a fractal dimension that approximates a pre-defined required value. For many dimensions, this is a non trivial task to be performed manually. The procedure we are proposing here closely parallels biological evolution, because it acts through three different levels: a genotype (a vector of integers), a protein-like intermediate level (the Lindenmayer Grammar) and a phenotype (the fractal curve). Variation acts at the genotype level, while selection is performed at the phenotype level (by comparing the dimensions of the fractal curves to the desired value).*

## **Introduction**

### **Fractals**

Some interesting geometrical questions were proposed and discussed during the last years of the nineteenth century. In 1890, Giuseppe Peano defined a curve that solved the following problem: "Is it possible for a curve to fill a square?" Several sets as odd as this curve, that seemed to be unclassifiable monsters, were formally studied. As a consequence of these works, the concept of classic dimension was revisited.

In 1919, the mathematician H.Hausdorff proposed a new definition of dimension, to be applied to distinguish these dubious cases from typical lines and surfaces. According to his definition, further refined by A.S. Besicovitch, monstrous curves may have a fractional dimension that to some extent measures the ratio between how much the curve grows in length and how much it advances.

In 1975 Mandelbrot [1] coined the term fractal to describe an heterogeneous class of sets that share some (not necessarily all) curious properties, such as self-similarity (the same shapes are found at different levels with different scales all over the set), underivability at every point, infinite length covered in a finite space, etc. All these sets have a fractional Hausdorff-Besicovitch dimension (see [1], chapter 39).

There are three main classes of fractal sets. Some appear as the boundary between convergence and

divergence of certain recursive mathematical functions in the complex domain; others are generated by means of random Brownian movements; the third class includes those curves obtained when a recursive transformation (iterator) is applied to an initial shape (initiator). Peano monstrous curve is in the latter group.

Fractals have been used for hundreds of applications in physics, chemistry, astronomy, geology, image compression, psychology, economics, medical imaging, et cetera. Many natural phenomena are better described using a fractional dimension. Fractals are thus used as descriptive models for the growth of plants, particle aggregation, river cartography, realistic images, and similar phenomena. Their fractal dimension characterizes most of these fractal models.

In physical systems, the fractal dimension reflects some properties of the system [2]. The physical characteristics of some bodies are related to the fractal dimension of their surfaces: the growth pattern of bacteria has a fractal dimension of 1.7. Another example is geological patterns: the fractal dimension of clouds is 1.30-1.33; 1.7 for snowflakes; 1.05-1.25 for coastlines in South Africa or Britain; 1.28-1.90 for woody plants and trees; et cetera. [3].

In medicine, fractal dimensions have been found for various biomolecules, such as DNA and proteins. For instance, the fractal dimension of Lysozyme (egg-white) is 1.614, for hemoglobin it is 1.583, for myoglobin 1.728 [4]. The fractal dimension of the perimeter of surface cell sections has been used to distinguish healthy cells and cancerous cells [5]. In analytical chemistry, the fractal dimension is used as a tool to characterize chemical patterns and problems of sample homogeneity [6]. A given fractal dimension makes it possible to simulate a variety of systems: fluid extraction or contaminant mitigation techniques [2], the hybrid orbital model of proteins [7], or the growth of conflict rate in aircraft flays [8].

Antennae are electromagnetic devices designed to radiate or capture signals. Some of their characteristics are gain, bandwidth, return loss and resonant frequencies. In the last years, fractal geometry has provided a new approach to traditional antenna design methods [9]. Several classical fractals of the initiator-iterator kind (von Koch's Snowflake, Sierpinski's gasket, for example) have been proposed as antenna prototypes. Certain properties of fractal antennae are related to its fractal dimension: an increase in the fractal dimension may be translated into higher gain, low return loss and a shifting down of the resonant frequencies. The algorithm described in this paper, that builds initiator-iterator fractals with a given fractal dimension, could be an interesting tool in the design, analysis or simulation of fractal antennae.

### ***Lindenmayer Grammars***

In 1968, Aristid Lindenmayer [10] defined a new class of grammars [11] (Lindenmayer systems or grammars, or L systems in short), similar to Chomsky grammars. Both kinds of grammars handle an initial string of symbols (the axiom) and include a set of production rules that may be applied to the symbols to generate new strings, but they differ in the way in which production rules are applied. Chomsky grammars change a symbol at a time sequentially, while Lindenmayer grammars apply many rules at the same time in parallel.

Let us look at an example of an L system: if we have the rules

A ::= B  
B ::= AB

and start at the word A, we get the following successive derivations:

A → B → AB → BAB → ABBAB → BABABBAB ...

This basic scheme is called a D0L system (a deterministic, context free, Lindenmayer system). There are different kinds of L systems that extend this scheme in different ways, but they are not the subject of this work.

D0L systems have been applied successfully to simulate different biological processes and to represent complex systems such as fractal curves [12] [13] [14], cellular automata [15] and others [16] [17].

### ***Fractal Representation by means of Lindenmayer Grammars***

Lindenmayer grammars provide a powerful tool to represent fractals of the recursive transformation type, such as Peano monstrous curve. The recursive transformation may be easily represented by means of a production rule, the initial shape by the axiom of the L system. The fractal curve is obtained from the series of words derived from the axiom by applying a *graphic representation scheme*. One of two main schemes are usually applied: vector graphics (associating a fixed vector displacement to each symbol in the L system alphabet), or turtle graphics, where the letters are interpreted as the movements in the graphic space of a turtle that remembers its current position and preceding direction. We have proved [14] that both schemes are equivalent for an interesting set of fractal curves.

As an example, let us consider the D0L system defined by axiom F--F--F and the following set of rules:

F ::= F+F--F+F  
+ ::= +  
- ::= -

The first derivation obtained from the axiom is:

F--F--F → F+F--F+F--F+F--F+F--F+F--F+F

With subsequent derivations, we get successive approximations to a well-known fractal curve, von Koch's snowflake curve, one of the first fractal curves in the history of Mathematics. The curve may be drawn by applying to the derived strings the following turtle graphics interpretation:

- F moves the turtle one step forward in its current direction.
- + increases by 60 degrees the current angle of the turtle direction.

- - decreases by 60 degrees the current angle of the turtle direction.

Figure 1 shows the graphical representation of the fifth derivation in the preceding L system.

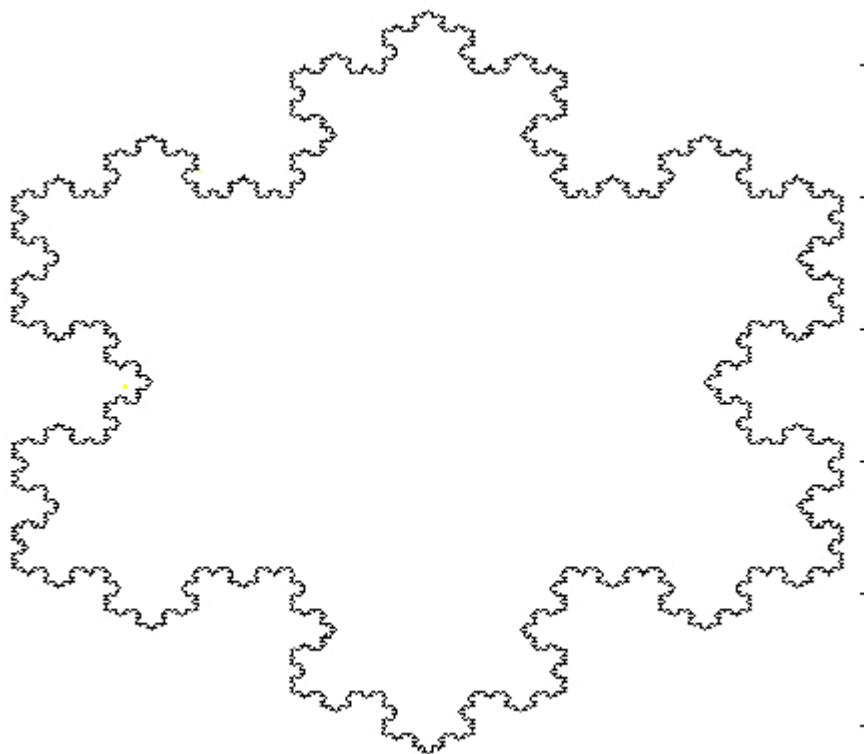


Figure 1. An approximation to von Koch's snowflake curve.

### ***Determination of fractal dimensions from equivalent L systems***

In previous works [18] [19], we have described an algorithm that estimates the fractal dimension of a non trivial subset of fractals of the recursive transformation type, by means of the equivalent L system. The dimension is easily determined by means of symbol manipulation, without using graphical procedures. For example, if our algorithm is applied to the L system representing von Koch's snowflake, the fractal dimension obtained is 1.2618595071429..., its Hausdorff-Besicovitch dimension. With this algorithm, we can choose to take into account the fact that the curve generated by the L system may overlap itself, which would change the actual dimension computed, depending of the definition used. This, however, increases the time required to compute the dimensions, and we have decided not to consider possible overlapping in the experiments described in this paper. A consideration of overlapping would not affect much the results described here, except by the corresponding increase in execution time and/or numbers of generations needed to reach the target.

Generating fractal curves of a given dimension can be done by means of deterministic techniques. Several tools use some measure of the regularity of continuous real functions with a single real variable. Hölder exponent is one of them [20]. In [21] three different methods are described to build a function that interpolates a set of points with a prescribed local regularity measured by Hölder exponent: by means of Schauder basis [22], using Weierstrass type functions and by a generalization of Iterated Function Systems (IFS).

Our method works non-deterministically on a formal representation of the target system, rather than the real curve. This approach seems more flexible and general, because formal models as Lindenmayer grammars are powerful enough to simulate a wide range of different complex systems. We hope that our technique is also applicable to other domains that can be described in this way.

### ***Grammatical Evolution***

Grammatical Evolution [23]-[30] is a grammar based, linear genome system, which has been applied in the area of Automatic Programming to automatically generate programs or expressions in a given language that solve a particular problem. Programming languages can be represented usually by context free Chomsky languages. In Grammatical Evolution, the Backus Naur Form (BNF) specification of a language is used to describe the output produced by the system (a compilable code fragment). Different BNF grammars can be used to automatically produce code in any language.

In Grammatical Evolution, the genotype is a string of 8 bit binary numbers generated at random, treated as integer values from 0 to 255. The phenotype is a running computer program generated by a genotype-phenotype mapping process. The mapping benefits from genetic code degeneracy, i.e. different integers in the genotype generate the same phenotype. According to Kimura's neutral theory [31], genetic code degeneracy maintains genotype diversity and enforces the preservation of valid phenotypes from run to run of the genetic engine.

When the string of integers in the genotype is exhausted before the phenotype has been completely generated, a biologically inspired wrapping mechanism is used to reuse the integers, similar to the gene-overlapping phenomenon observed in many organisms in nature. The genotype-phenotype mapping in Grammatical Evolution is deterministic; i.e. each individual is always mapped to the same phenotype. Two mechanisms are used to minimize the numbers of invalid individuals in each generation: punishing them with poor fitness values, or using a steady state replacement method [29]. The last method seems to improve greatly the performance of the algorithm.

In Grammatical Evolution, standard genetic algorithms are applied to the different genotypes in a population, using the typical crossover and mutation operators. For each domain, one must design the proper fitness function, which will be used by the genetic algorithm to perform selection. This technique has been successfully applied to the automatic programming of problems in different domains: symbolic regressions, finding trigonometric identities, the Santa Fe ant trail, and caching algorithms.

This paper extends Grammatical Evolution to L systems, to solve the problem of obtaining arbitrary fractal curves with a given dimension. The same approach could be used in other domains. Previous work by other authors has applied genetic algorithms to L systems, rather than Grammatical Evolution. Ochoa [32] evolves D0L systems with a single rule, that generate shapes similar to plants. Other authors [33][34][35] evolve parametric L systems (an extension of Lindenmayer grammars) [36], which means that they are faced with the important problem that parametric systems are not closed under the action of genetic algorithms. This problem would be solved easily by our approach, because Grammatical Evolution goes through an intermediate grammar (in this case it would describe a valid parametric system), which ensures that the actual L system generated is syntactically correct. In this

paper we are not tackling this problem and use D0L systems, where simple genetic algorithms would be sufficient. However, we have grounds to prefer the Grammatical Evolution approach, because it allows us to evolve at the same time the L system and the angle used in its graphic interpretation, as described later.

### ***The design of L systems that represent curves with a given fractal dimension***

Designing fractal curves with a given dimension is relatively easy for certain values of the desired dimension, but very difficult for others. The following L system rules represent (with a turtle graphic interpretation based on an angle step of 60 degrees) the iterators for three different fractal curves with the same dimension: 1.2618595... ( $\log 4 / \log 3$ ). The first one, as shown above, corresponds to von Koch's snowflake curve. All four, and a few more, could have been obtained by hand, by a simple geometrical study of the curve iterator.

```
F ::= F+F--F+F
F ::= F+F-
F ::= +F--FF-F+
F ::= F+F-F-F+
```

On the other hand, designing a fractal curve with a dimension of 1.255 would be much more complicated. The first step would consist of obtaining two integer numbers, *a* and *b*, such that

$$1.255 = \log a / \log b$$

This step could be relaxed to asking for two integers such that the given dimension would be approximated within some degree of accuracy (for instance, 0.001).

The second step would be to design a geometrical iterator such that it would take *a* steps to advance a distance equal to *b*.

We solve this problem automatically by means of Grammatical Evolution. Our genetic algorithm acts on genotypes made of vectors of integers, and makes use of a fixed grammar to translate the genotypes into an intermediate level, which can be interpreted as a rule for an L system which, together with a turtle graphic interpretation, generates the final phenotype: a fractal curve with the desired dimension, or an approximation of the same.

### ***The developmental algorithm***

The initial population consists of 64 vectors of 8 integers in the interval [0,10]. Vectors of different lengths are later generated by the genetic algorithm. Other intervals (such as [0,255]) can be used, so as to include genetic code degeneracy. This has been tested and it also works, although no significant improvement in performance has been detected.

In our first experiment, the genotype of one individual in the population (a vector of *n* integers) is translated by making use of the following D0L grammar:

```
0: F ::= F
1: F ::= FF
2: F ::= F+
3: F ::= F-
4: F ::= +F
5: F ::= -F
6: F ::= F+F
7: F ::= F-F
```

8: F ::= +  
 9: F ::= -  
 10: F ::= e

where e is the empty string.

The translation is performed according to the following developmental algorithm:

1. The axiom (first word) of the D0L grammar is assumed to be F.
2. As many elements from the remainder of the genotype are taken (and removed) from the left of the genotype as the number of F in the current word. If there remain too few elements in the genotype, the required number is completed circularly.
3. The current word derives a new one in the following way: each F in the word is replaced by the right hand side of the rule with the same number as the integers obtained by the preceding step. In the case of genetic code degeneracy, the remainder of the integers modulo 11 is used.
4. If the genotype is now empty, the algorithm stops, and the last derived word is the output.
5. If the derived word has no F, the whole word is replaced by the axiom.
6. Go to step 2.

In any derivation, the following implicit rules are also applied:

+ ::= +  
 - ::= -

Let us look at an example. Let the individual genotype to be translated be the 7-element vector

10 6 7 6 0 2 7

We start from axiom F. It contains one F. Therefore, at step 2 we extract one element from the left of the genotype (10). The remainder of the genotype becomes

6 7 6 0 2 7

In step 3, by applying rule 10, the axiom derives e (the empty string). The derived word has no F, thus in step 5 we replace it by the axiom F. This is the second word in the derivation. We go back to step 2. The current word contains one F. Therefore, we take one element (6) from the remainder of the genotype, which becomes

7 6 0 2 7

In step 3 we now apply rule 6 to the only F, deriving F+F. This is the third word in the derivation. We go back to step 2. The current word contains two F, therefore, we take two elements (7,6) from the remainder of the genotype, which becomes

0 2 7

We now apply rule 7 to the first F and rule 6 to the second F in F+F, deriving F-F+F+F. This is the fourth word in the derivation. We go back to step 2. The current word contains four F. Therefore, we should take four elements from the remainder of the genotype, but we only have three. We complete the required number circularly and take (0,2,7,0). The genotype vector becomes empty.

We now apply rule 0 to the first F, rule 2 to the second, rule 7 to the third and rule 0 to the fourth F in



F-F+F+F, deriving F-F++F-F+F. This is the last word in the derivation, the result of the algorithm.

We can now simplify the output by erasing unnecessary +- pairs, if any (there are none in this case). We also may add or delete + or - signs at the beginning and the end of the word, so that the turtle ends its movement in the same direction it started (this is a requirement for some of the theorems we are applying). In this case, we get F-F++F-F+F-. The rules of the D0L system generated by the developmental algorithm are:

```
F ::= F-F++F-F+F-  
+ ::= +  
- ::= -
```

### ***The genetic algorithm***

We can now apply the algorithm described in [18], [19] to compute from F-F++F-F+F- the dimension of the fractal curve obtained from the D0L system by means of a turtle graphic interpretation with a given angle step. This dimension can be compared to the target dimension, providing a fitness rule for the genetic algorithm.

The scheme for the genetic algorithm is as follows:

1. Generate a random population of 64 vectors of 8 integers in the [0,10] or the [0,255] interval.
2. Translate every individual genotype into a word in the alphabet {F+-}, using the developmental algorithm described above.
3. Compute the dimension of the fractal curve represented by the corresponding D0L system.
4. Compute the fitness of every genotype as  $1/|\text{target-dimension}|$ .
5. Order the 64 genotypes from higher to lower fitness.
6. If the highest fitness genotype has a fitness higher than the target fitness, stop and return this genotype.
7. From the ordered list of 64 genotypes created in step 5, remove the 16 genotypes with least fitness (leaving 48) and take the 16 genotypes with most fitness. Pair these 16 genotypes randomly to make 8 pairs. Each pair generates another pair, a copy of their parents, modified according to four genetic operations. The new 16 genotypes are added to the remaining population of 48 to make again 64, and their fitness is computed as in steps 2 to 4.
8. Go to step 5.

The four genetic operations mentioned in the algorithm are:

- Recombination (applied to 100% generated genotypes). Given a pair of genotypes,  $(x_1, x_2 \dots x_n)$  and  $(y_1, y_2 \dots y_m)$ , a random integer is generated in the interval  $[0, \min(n,m)]$ . Let it be  $i$ . The resulting recombined genotypes are:  $(x_1, x_2 \dots x_{i-1}, y_i, y_{i+1} \dots y_m)$  and  $(y_1, y_2 \dots y_{i-1}, x_i, x_{i+1} \dots x_n)$ .
- Mutation (applied to  $n1$  % generated genotypes if both parents are equal, to  $n2$  % if they are different). It consists of replacing a random element of the vector by a random integer in the same interval.
- Fusion (applied to  $n3$  % generated genotypes). The genotype is replaced by a catenation of itself with a piece randomly broken from either itself or its brother's genotype. (In some tests, the whole genotype was used, rather than a piece of it).
- Elision (applied to 5% generated genotypes). One integer in the vector (in a random position) is eliminated.

The last two operations allow longer or shorter genotypes to be obtained from the original 8 element vectors. The optimal values of n1 (100), n2 (100) and n3 (25) have been obtained by means of a set of 22 tests that combine different angles and target dimensions. Table 1 shows that these parameters are important, for different combinations of values give rise to very different computing times.

n1	n2	n3	Average generations	Average CPU time
20	20	5	6668	1838
50	20	5	2979	1888
50	50	5	3794	3211
80	10	5	2625	1590
80	80	5	3917	1430
100	100	5	2216	1007
100	100	1	10172	4776
100	100	10	1027	615
100	100	25	146	176
100	100	50	163	497
100	100	90	49	497

Table 1: Results of experiments to get optimal values of genetic operation rates.

The algorithm has three input parameters: the target dimension, the target minimum fitness, and the angle step for the turtle graphics interpretation.

This procedure is similar to biological evolution in many respects. There are three different levels (see figure 2):

1. The genotype (nucleic acids), here represented by vectors of integers.
2. The intermediate level (proteins), here represented by words on the {F,+,-} alphabet. The translation from the genotype to the intermediate level is performed using a fixed grammar (the equivalent of the fixed genetic code).
3. The final phenotype (organisms), here represented by the fractal curves, which are obtained from the L systems built from the intermediate level words by means of a turtle graphic interpretation.

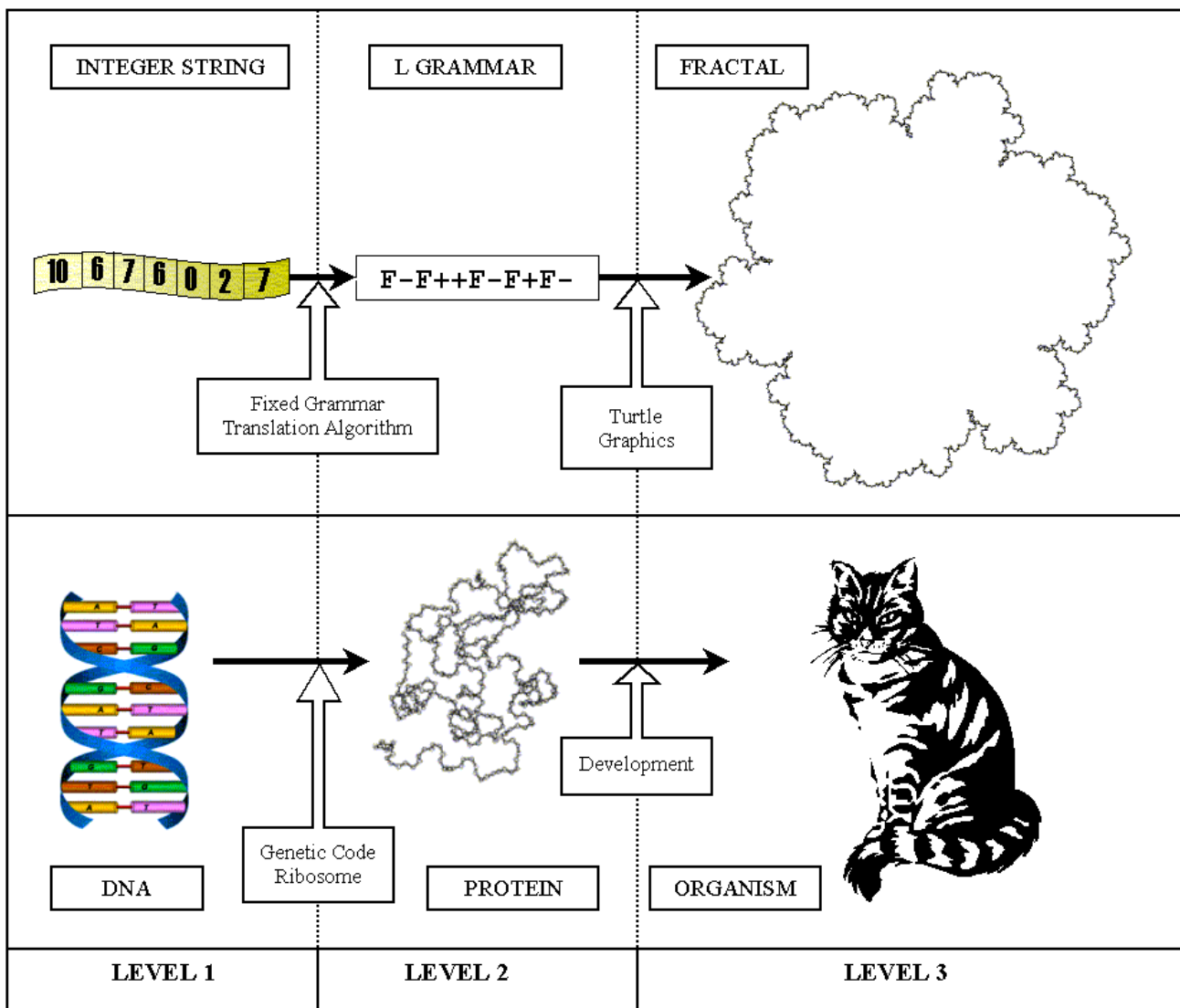


Figure 2. Parallels between our Grammatical Evolution approach and biological evolution.

In a second experiment, the genotype of each individual in the population contains one more element (it is a vector of  $n+1$  integers). The first element (or its remainder modulo 11) is interpreted as an index to a vector that defines the angle to be used in the graphic interpretation of the phenotype. Eleven possible angles have been used: 120, 90, 72, 60, 45, 40, 36, 30, 24, 20 and 18 degrees (i.e. the first submultiples of 360). The developmental algorithm is applied only to the last  $n$  elements of the genotype. The genetic algorithm applies to all the  $n+1$  elements of the genotype. In this way, the angle itself evolves, and fractal curves with unexpected angles may be obtained.

### Results

The algorithm described above reaches its targets with surprising speed. Sometimes (for the simplest dimensions, those that can be done by hand) the target is reached in the first generation: in a set of 64 random eight-element genotypes, there is a big probability of having the codification of one of those phenotypes). For other, less standard dimensions, the number of generations to reach a given approximation to the target is usually larger, sometimes quite large. Table 2 shows a few of the results

we have obtained.

Dimension	Angle	Nr.of tests	Number of generations to reach target
1.1	45	10	37 to 9068
1.1	60	4	119 to 72122
1.2	45	8	188 to 11173
1.2	60	10	21 to 750
1.3	45	9	50 to 18627
1.3	60	4	14643 to 66274
1.25	60	2	1198 to 3713
1.255	60	15	1 to 2422
1.2618595...	60	4	1 to 2
1.4	45	10	79 to 781
1.4	60	10	33 to 1912
1.5	45	11	52 to 11138
1.5	60	8	12 to 700
1.6	45	5	275 to 3944
1.6	60	1	116913
1.7	45	2	585 to 1456
1.7	60	8	18 to 1221
1.8	45	2	855 to 2378
1.8	60	13	69 to 3659
1.9	72	1	5467
1.95	90	1	956
2	45	5	1
2	90	5	1

Table 2. Number of generations to reach the target in a set of tests of our Grammatical Evolution approach.

Since the algorithms use random numbers, different random seeds give different results. We have thus obtained sets of fractal curves, sometimes quite different in appearance, that share the same fractal dimension. Table 3 shows some results for a target dimension of 1.255 and an angle of 60 degrees. In all of them, the minimum fitness was set to 1000 (which corresponds to an error in the target dimension below 0.001). The dimension of all the results came to be 1.2549. This fractal dimension has been computed without considering possible overlappings of the curves with themselves. A definition of dimension that would take this into account could also be considered [18], [19], at the cost of longer computation times, and perhaps more generations. Figure 3 displays the fractal curves, approximated by the fourth derivation of the corresponding L systems.

Number of generations	Size of genotype	L System word developed	Axiom
4	16	-F+FF+FF-	F--F--F
44	7	F-F++F-F+F-	F++F++F
72	8	FF--F+FF+	F++F++F
255	8	FF-FF++F-	F--F--F

Table 3. Different fractal curves sharing the same dimension, evolved by our method.

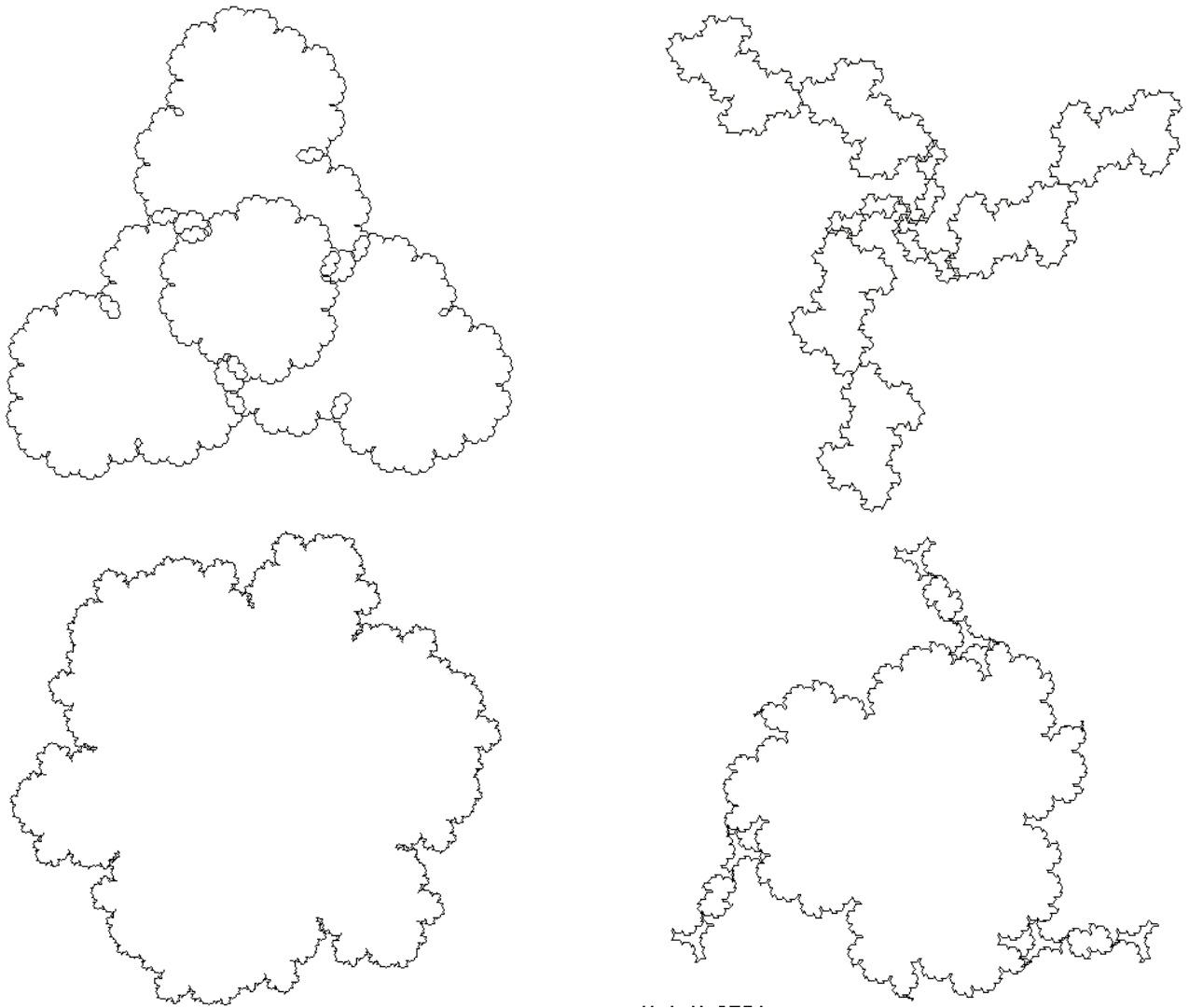


Figure 3. Four different fractal curves evolved for a target dimension of 1.255

Table 4 shows a few interesting fractals evolved by means of our algorithms. The first one has the same dimension as von Koch's snowflake, with an angle of 36 degrees. Figures 8 to 13 display the fractal curves, approximated by the third derivation of the corresponding L systems.

Dimension	Angle	Number of generations	Size of genotype	L System word developed	Axiom	Shown in figure
1.26178...	36	27	35	+F+F---F-F+F- F+F+	F-F-F-F-F- F-F-F-F-F	4
1.5018	45	2000	17	++F-F--F+F+F +FF---F-F++	F---F++F--- F-F--F	5
1.8998	40	1460	41	++F-FF-F-F--- F-F---F--F- F++	F+F+F+F+F+F +F+F+F	6
1.8008	45	2378	30	F-F-F-FF-F-F- -F-F--F-F---F -FF-F-F+F+FF-	F---F++F--- F-F--F	7
1.7005	60	18	22	++FF+F+FFF+F+ F+FFF+FFF+F+F -F+F----	F-F-F-F-F-F	8
1.6006	45	275	30	+F-F--F-- F+FF--F+F-F- FF+FF-F--F	F++F++F++F	9

Table 4. A few fractal curves evolved by our method.

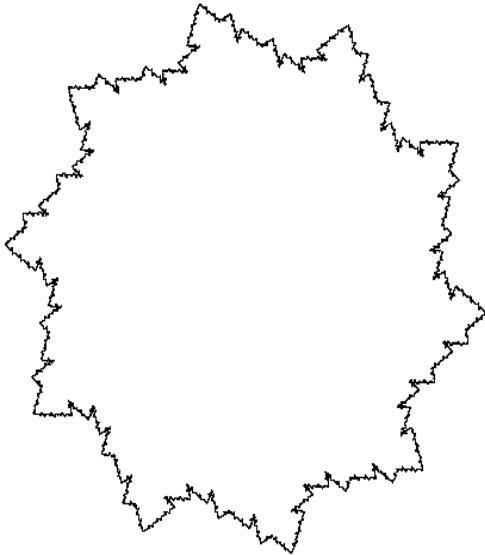


Figure 4. A fractal curve with the same dimension as von Koch's snowflake.

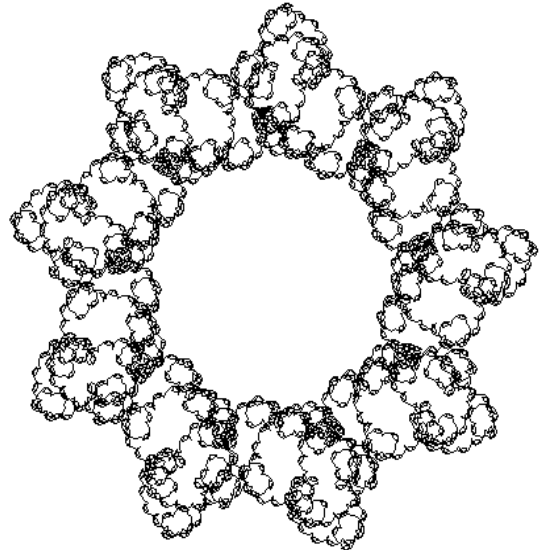


Figure 6. A fractal curve with approximate dimension 1.9.

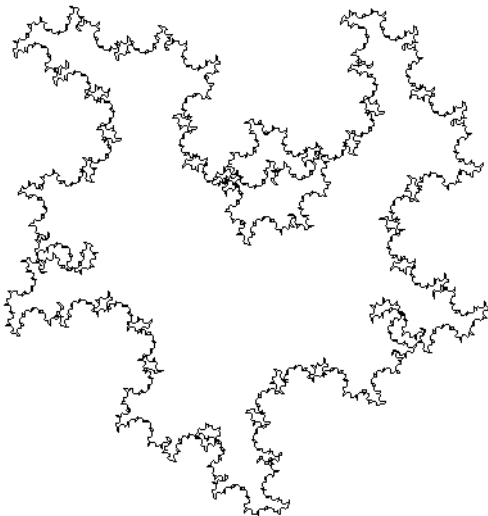


Figure 5. A fractal curve with approximate dimension 1.5.

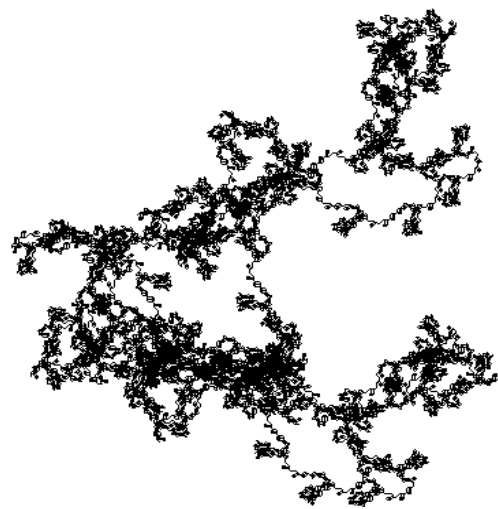


Figure 7. A fractal curve with approximate dimension 1.8.

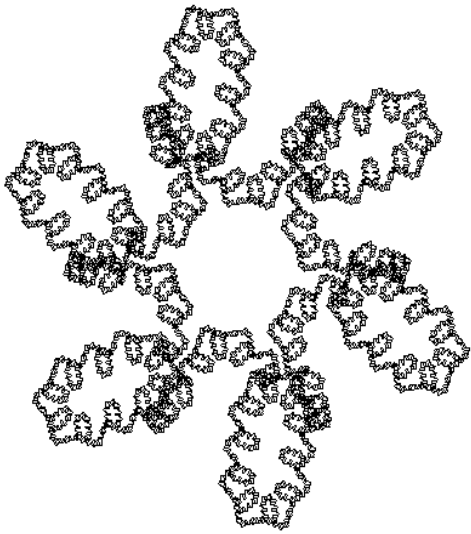


Figure 8. A fractal curve with approximate dimension 1.7.

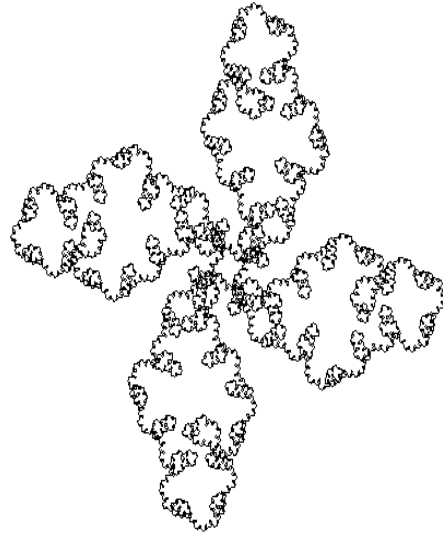


Figure 9. A fractal curve with approximate dimension 1.6.

Table 5 shows finally some of the results obtained with our second experiment, where the turtle angle itself was subject to evolution by means of the genetic algorithm.

Target dimension	Actual dimension	Number of generations	Size of genotype	Angle evolved
1.5	1.5	5	8	90
1.5	1.5	8	14	45
1.5	1.4999	64	19	90
1.5	1.4999	176	21	90
1.5	1.5	50	9	90
1.5	1.5009	940	124	36
1.5	1.5008	1337	86	36
1.5	1.5	68	9	90
1.9	1.8992	1069	55	72
1.9	1.8993	1306	42	24
1.9	1.8998	1460	41	40

Table 5. A set of tests where the turtle angle was evolved too.

### **Conclusions and future research lines**

Grammatical Evolution has been applied to generate and evolve Lindenmayer Grammars that represent fractal curves with a pre-defined fractal dimension. The procedure described parallels biological evolution by acting through three different levels: a genotype (a vector of integers), a protein-like intermediate level (the Lindenmayer Grammar) and a phenotype (the fractal curve). Variation acts at the genotype level, while selection is performed at the phenotype level (by comparing the dimensions of the fractal curves to the desired value).

The results show the power of the approach. Those cases where a solution is easily found by hand were also easily solved by our algorithms. Many interesting curves have been found in more difficult cases: this paper shows a few of them. Evolution towards the target was relatively fast, at least with our choice of values for the parameters of the algorithms.

The angle used for the graphical interpretation of the fractal curve has been introduced as a pre-defined parameter in some experiments and automatically calculated in others. The latter approach generated a

few good cases with unusual angles, although it has been noticed that variations in the angle tend to be favorable only during the first generations: once a good angle has been evolved, it is not easily changed, because changing the angle of the graphical interpretation has a massive effect on the dimension of the curve.

Similar results could have been obtained by directly applying genetic algorithms on L systems, represented by arbitrary strings with characters 'F', '+' and '-'. We have actually performed this experiment, and found no significant differences in number of generations or execution time, as compared to the Grammatical Evolution approach. However, the latter is much better if the angle is also evolved, as described in the previous paragraph. It would also present important advantages if used with parametric L systems.

In the future, we plan to work on the following questions:

- Evolution of other fractal properties besides dimension.
- Application of our approach to solve problems in other areas where L Systems are applicable.
- Its application to parametric L systems.

### **Acknowledgments**

This paper has been sponsored by the Spanish Ministry of Science and Technology (MCYT), project numbers TIC2002-01948 and TIC2001-0685-C02-01.

### **References**

- [1] B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H.Freeman and Company, New York (1977).
- [2] K. Mela, J.N. Louie, "Correlation length and Fractal dimension interpretation from seismic data using variograms and power spectra," *Geophysics*, **66**, No. 5, 1372-1378 (2001).
- [3] R.P. Taylor, B. Spehar, C.W.G Clifford and B.R Newell, "The Complexity of Pollack's Dripped Fractals" *Proceedings of the International Conference of Complex Systems* (2002).
- [4] P.M.Iannaccone and M.Khokha eds., *Fractal Geometry in Biological Systems: An Analytical Approach*, CRC Press, Boca Raton, Fla.1996.
- [5] W. Bauer and C.D. Mackenzie, "Cancer Detection via determination of fractal cell dimension," *Workshop on computational and theoretical biology* (1999).
- [6] K. Danzer, J.F. van Staden and D.T. Burns, "Concepts and Applications of the Term 'Dimensionality' in Analytical Chemistry", *Pure Applied Chemistry*, **74**, No. 8, 1479-1487 (2002).
- [7] F.Torrens, "Fractals Hybrid orbitals in Protein Models," *Complexity International*, **8** (2001).
- [8] S. Mondoloni and D. Liang, "Airspace Fractal dimension and applications," *3<sup>rd</sup> USA/Europe ATM R& D Seminar* (2001).
- [9] K.J.Vinoy, K.A.Jose, V.K.Varadan and V.V.Varadan, "Hilbert Curve Fractal Antenna: a Small Resonant Antenna for VHF/UHF Applications," *Microwave and Optical Technology Letters*, **29** No.4, 215-219 (2001).
- [10]A. Lindenmayer, "Mathematical Models for Cellular Interactions in Development" (two parts), *J.Theor. Biol.*, **18**, 280-315 (1968).
- [11]G. Herman and G. Rozenberg, *Developmental systems and languages*. North-Holland/American Elsevier, Amsterdam (1975).
- [12]E. G. Giessmann, "Generation of fractal curves by generalization of Lindenmayer's L Systems".



- Proceedings of the 1<sup>st</sup>. IFIP Conference on Fractals in the Fundamental and Applied Sciences*. H.-O. Peitgen, J. M. Henriques, and L.F. Penedo, Editors, North-Holland, Amsterdam, 147-157 (1991).
- [13]P. Prusinkiewicz, "Graphical Applications of L-Systems". *Proceedings of Graphical Interface 86 and Vision Interface 86*. M. Wein and E. M. Kidd, Editors. Vancouver, BC, 247-253 (1986).
- [14]M. Alfonsoeca and A. Ortega: "A study of the representation of fractal curves by L systems and their equivalences", *IBM Journal of Research and Development*, **41**, No. 6, 727-736 (1997).
- [15]M. Alfonsoeca and A. Ortega: "Representation of some cellular automata by means of equivalent L systems", *Complexity International*, ISSN: 1320-0682, **7**, 1-16 (2000), <http://www.csu.edu.au/ci/vol07/alfons01/>
- [16]J. D. Corbit and D. J. Garbary, "Computer simulation of the morphology and development of several species of seaweed using Lindenmayer systems". *Computers & Graphics*, **17**, No. 1, 85-8 (1993).
- [17]G. Rozenberg and A. Salomaa, (Editors) *Lindenmayer Systems. Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*. Springer-Verlag, Berlin (1992).
- [18]M. Alfonsoeca and A. Ortega: "Using APL2 to Compute the Dimension of a Fractal Represented as a Grammar", *APL Quote Quad*, **30**, No. 4, 13-23 (2000).
- [19]M. Alfonsoeca and A. Ortega: "Determination of fractal dimensions from equivalent L systems", *IBM Jr. of Res. and Dev.* **45**, No. 6, 727-736 (2001).
- [20]B.Guiheneuf, S.Jaffard and J.Levy Veהל, "Two Results Concerning Chirps and 2-microlocal Exponents Prescription", *Applied and Computational Harmonic Analysis*, **5**, 487-492 (1998).
- [21]K.Daoudi, J.Levy Veהל and Y.Meyer, "Construction of continuous functions with prescribed local regularity", *Journal of Constructive Approximation*, **14**, No. 3 349-385 (1998).
- [22]S.Jaffard. "Functions with Prescribed Hölder Exponent", *Applied and Computational Harmonic Analysis*, **2**, No. 4, 400-401 (1995).
- [23]M. O'Neill and C. Ryan: "Grammatical Evolution". *IEEE Transactions on Evolutionary Computation*. **5**, Np.4, 349-358 (2001).
- [24]M. O'Neill and C. Ryan: "Evolving Multi-line Compilable C Programs". In *Proceedings of the Second European Workshop on Genetic*, Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science **1598**, 83-92 (1999).
- [25]M. O'Neill and C. Ryan: "Under the Hood of Grammatical Evolution". In *GECCO '99: Proceedings of the Genetic and Evolutionary Computation Conference 1999*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. San Mateo, CA: Morgan Kaufmann, **2**, 1143-1148 (1999).
- [26]M. O'Neill and C. Ryan: "Genetic code degeneracy: implications for grammatical evolution and beyond". In *ECAL'99: Proceedings of the Fifth European Conference on Artificial Life*, Lausanne, Switzerland, 149-153 (1999).
- [27]C. Ryan, J.J. Collins, and M. O'Neill: "Grammatical Evolution: Evolving Programs for an Arbitrary Language". In *EuroGP'98: Proceedings of the First European Workshop on Genetic Programming*. Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science **1391**, 83-95 (1998).
- [28]C. Ryan, O'Neill M. "Grammatical Evolution: A Steady State Approach". In *Proceedings Joint Conference on Information Sciences*, Research Triangle Park, NC, 419-423 (1998).
- [29]C. Ryan, M. O'Neill: "Grammatical Evolution: A Steady State Approach". In *Genetic Programming: Proceedings of the 3<sup>rd</sup> Annual Conference*, J. R. Koza, W. Banzhaf, L. Chellapilla, K. Deb, M. Dorigo, D. B.Fogel, M. H. Garzon, D. E. Goldberg, H. Iba and R. L. Riolo, Ed.s Cambridge, MA, 180-185 (1998).

- [30]C. Ryan, M. O'Neill, and J. J. Collins: "Grammatical Evolution: Solving Trigonometric Identities." In Mendel'98: Proceedings of the 4<sup>th</sup> International Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, and Rough Sets. Brno, Czech Republic: Tech. Univ. Brno, 111-119 (1998).
- [31]M. Kimura: *The neutral theory of molecular evolution*. Cambridge University Press (1983).
- [32]G.Ochoa: "On Genetic Algorithms and Lindenmayer Systems." Proc. PPSN IV, Amsterdam, Springer-Verlag, Lecture Notes in Computer Science **1498**, 335-343 (1998).
- [33]G.S.Hornby: *Generative Representations for Evolutionary Design Automation*. A dissertation presented to the Faculty of the Graduate School of Arts and Sciences, Brandeis University Department of Computer Science (advisor: Jordan B. Pollack), in partial fulfillment of the requirements for the Ph.D. degree (2003).
- [34]C.Jacob: "Genetic L-System Programming." PPSN III – Parallel Problem Solving from Nature, International Conference on Evolutionary Computation, Springer-Verlag, Berlin, Lecture Notes in Computer Science **866**, 334 – 343 (1994).
- [35]C.Traxler, M.Gervautz: "Using Genetic Algorithms to Improve the Visual Quality of Fractal Plants generated with CSG-PL Systems." Proceedings of The Fourth International Conference in Central Europe on Computer Graphics and Visualization'96 Ed. N.Magnenat-Thalmann V.Skala, University of West Bohemia, Plzen, Czech Republic (1996).
- [36]A.Ortega, M.de la Cruz, M.Alfonseca: "Parametric 2-dimensional L Systems and recursive fractal images: Mandelbrot set, Julia sets and biomorphs", Computers and Graphics (Elsevier), **26** No. 1, 143-149, 2002.