Grammatical evolution to design fractal curves with a given dimension

A. Ortega
A. A. Dalhoum
M. Alfonseca

Lindenmayer grammars have frequently been applied to represent fractal curves. In this work, the ideas behind grammar evolution are used to automatically generate and evolve Lindenmayer grammars which represent fractal curves with a fractal dimension that approximates a predefined required value. For many dimensions, this is a nontrivial task to be performed manually. The procedure we propose 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 19th 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 that were as odd as this curve and 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 in order 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 B. B. Mandelbrot [1] coined the term *fractal* to describe a heterogeneous class of sets that share some (though 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, and infinite length covered in a finite space. All of 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; and the third class includes those curves obtained when a recursive transformation (iterator) is applied to an initial shape (initiator). The Peano monstrous curve is in the latter group.

Fractals have been used for hundreds of applications in fields such as physics, chemistry, astronomy, geology, image compression, psychology, economics, and medical imaging. Many natural phenomena are better described using a fractional dimension, and 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. For example, the growth pattern of bacteria has a fractal dimension of 1.7, and the fractal dimension of

©**Copyright** 2003 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

0018-8646/03/$5.00 © 2003 IBM

clouds is 1.30 to 1.33; for snowflakes it is 1.7, for coastlines in South Africa or Britain, 1.05 to 1.25, and for woody plants and trees, 1.28 to 1.90 [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, and for myoglobin 1.728 [4]. The fractal dimension of the perimeter of surface cell sections has been used to distinguish healthy cells from 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 flight schedules [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 methods of antenna design [9]. Several classical fractals of the initiator–iterator kind (for example, Von Koch's snowflake, Sierpinski's gasket) have been proposed as antenna prototypes. Certain properties of fractal antennae are related to their 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. In this paper we describe an algorithm that builds initiator–iterator fractals with a given fractal dimension, and thus 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, in short, *L systems*), 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

$$A ::= B$$

and

$$B ::= AB,$$

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

$$A \rightarrow B \rightarrow AB \rightarrow BAB \rightarrow ABBAB \rightarrow BABABBAB \ldots .$$

This basic scheme is called a *D0L system* (a deterministic, context-free L 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 [12–14], 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 the Peano monstrous curve. The recursive transformation may easily be 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 is usually applied: vector graphics (associating a fixed vector displacement with each symbol in the L-system alphabet), or turtle graphics, in which the letters are interpreted as the movements in the graphic space of an invisible "turtle" that remembers its current position and preceding direction. We have proved [14] that the two 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:

$$F ::= F + F--F + F,$$

$$+ ::= +,$$

and

$$- ::= -.$$

The first derivation obtained from the axiom is

$$F--F--F \rightarrow F + F--F + F--F + F--F + F--F.$$

With subsequent derivations, we obtain 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° the current angle of the turtle direction.
- − decreases by 60° the current angle of the turtle direction.

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

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 nontrivial 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, depending on the definition used, would change the actual dimension computed. However, this 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 have much effect on 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. The 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 the Hölder exponent: by means of the Schauder basis [22], using Weierstrass-type functions, and by a generalization of iterated function systems (IFS).

Our method works nondeterministically 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. It has been applied to automatic programming to automatically generate programs or expressions in a given language that solve a particular problem. Programming languages can usually be represented 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).

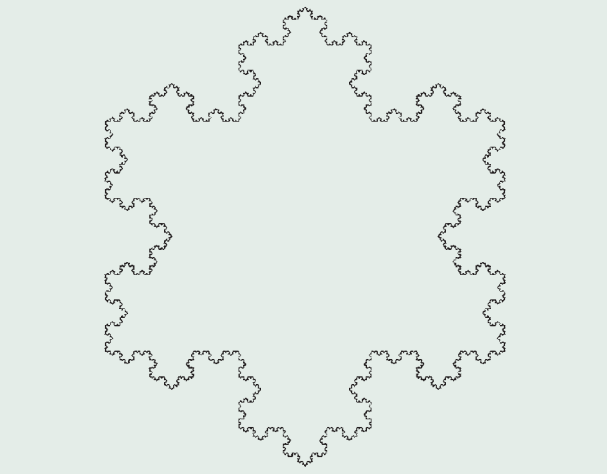

Figure 1

An approximation of Von Koch's snowflake curve.

Different BNF grammars can be used to produce code automatically in any language.

In grammatical evolution, the genotype is a string of eight-bit binary numbers generated at random and 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, similar to the gene-overlapping phenomenon observed in many organisms in nature, is employed to reuse the integers. 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 number of invalid individuals in each generation: punishing them with poor fitness values, or using a steady-state replacement method [29]. The latter method seems to improve the performance of the algorithm greatly.

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 that 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 generates shapes similar to plants. Other authors [33–35] evolve parametric L systems (an extension of Lindenmayer grammars) [36] and encounter the important problem that parametric systems are not closed under the action of genetic algorithms. This problem could easily be solved 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 do not address this problem. We 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 simultaneously evolve 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°) 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 through 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 consisting of vectors of integers. It 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 that, 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 eight 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. These have been tested and shown to work, 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 seven-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, so in step 5 we replace it with the axiom F. This is the second word in the derivation. We return 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 apply rule 6 to the only F, deriving $F + F$. This is the third word in the derivation. We return to step 2. The current word contains two Fs; 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 return to step 2. The current word contains four Fs; 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 may also 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 obtain $F-F++F-F+F-$. 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 with 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 eight 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 $1/|target\text{–}dimension|$.
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 eight 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 again make 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 the following:

- *Recombination* (applied to 100%-generated genotypes). Given a pair of genotypes, $(x_1, x_2, \cdots, x_n)$ and $(y_1, y_2, \cdots, y_m)$, a random integer is generated in the interval [0, min(n, m)]. Let it be i. The resulting

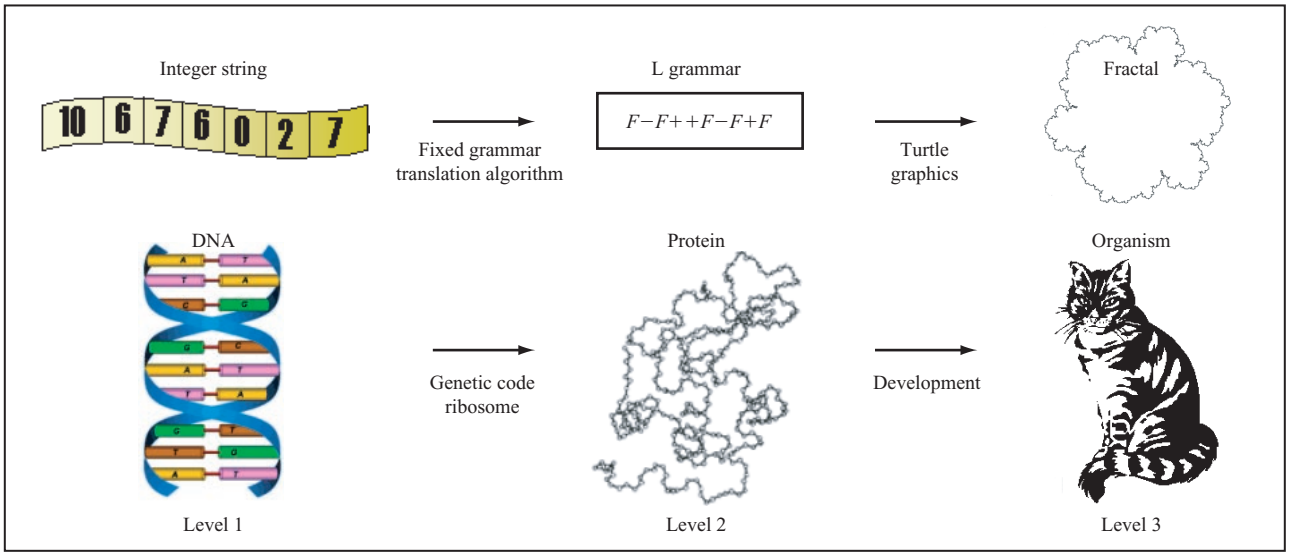

Figure 2

Parallels between our grammatical evolution approach and biological evolution.

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

n1	*n2*	*n3*	*Average generations*	*Average CPU time (s)*
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	10,172	4776
100	100	10	1027	615
100	100	25	146	176
100	100	50	163	497
100	100	90	49	497

recombined genotypes are $(x_1, x_2, \cdots, x_{i-1}, y_i, y_{i+1}, \cdots, y_m)$ and $(y_1, y_2, \cdots, y_{i-1}, x_i, x_{i+1}, \cdots, x_n)$.

- *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 with a random integer in the same interval.
- *Fusion* (applied to $n3\%$-generated genotypes). The genotype is replaced with 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 eight 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.

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 in many respects to biological evolution. There are three different levels (**Figure 2**):

1. The *genotype* (nucleic acids), here represented by vectors of integers.
2. The *intermediate* level (proteins), here represented by words using 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 that are obtained from the L systems built from the intermediate-level words by means of a turtle graphic interpretation.

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

Dimension	*Angle* (degrees)	*Number of tests*	*Number of generations to reach target*
1.1	45	10	37 to 9068
1.1	60	4	119 to 72,122
1.2	45	8	188 to 11,173
1.2	60	10	21 to 750
1.3	45	9	50 to 18,627
1.3	60	4	14,643 to 66,274
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 11,138
1.5	60	8	12 to 700
1.6	45	5	275 to 3944
1.6	60	1	116,913
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

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° (i.e., the first submultiples of 360°). The developmental algorithm is applied to only the last n elements of the genotype. The genetic algorithm applies to all of 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 high probability of encountering the codification of one of those phenotypes). For other, less standard dimensions, the number of generations required to reach a given approximation to the target is usually larger, and can sometimes be quite large. **Table 2** shows a few of the results we have obtained.

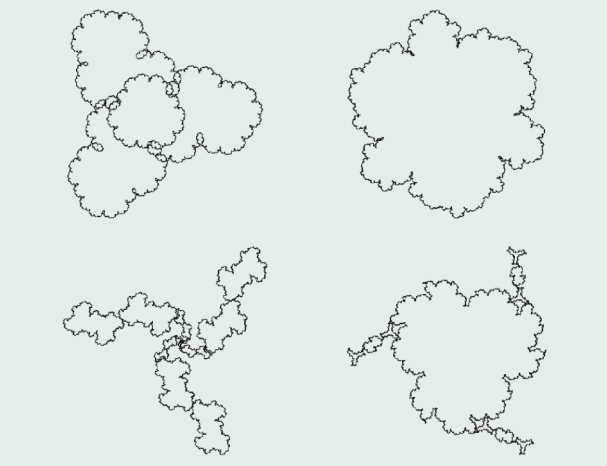

Figure 3

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

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

Number of generations	*Genotype size*	*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	$FF-FF++F-$	$F--F--F$

Because 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°. 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 of 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], but 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.

Figure 4 shows a few interesting fractals evolved by means of our algorithms. Figure 4(a) has the same dimension as Von Koch's snowflake, with an angle of 36°. Figures 4(b) to 4(f) display the fractal curves approximated by the third derivation of the corresponding L systems.

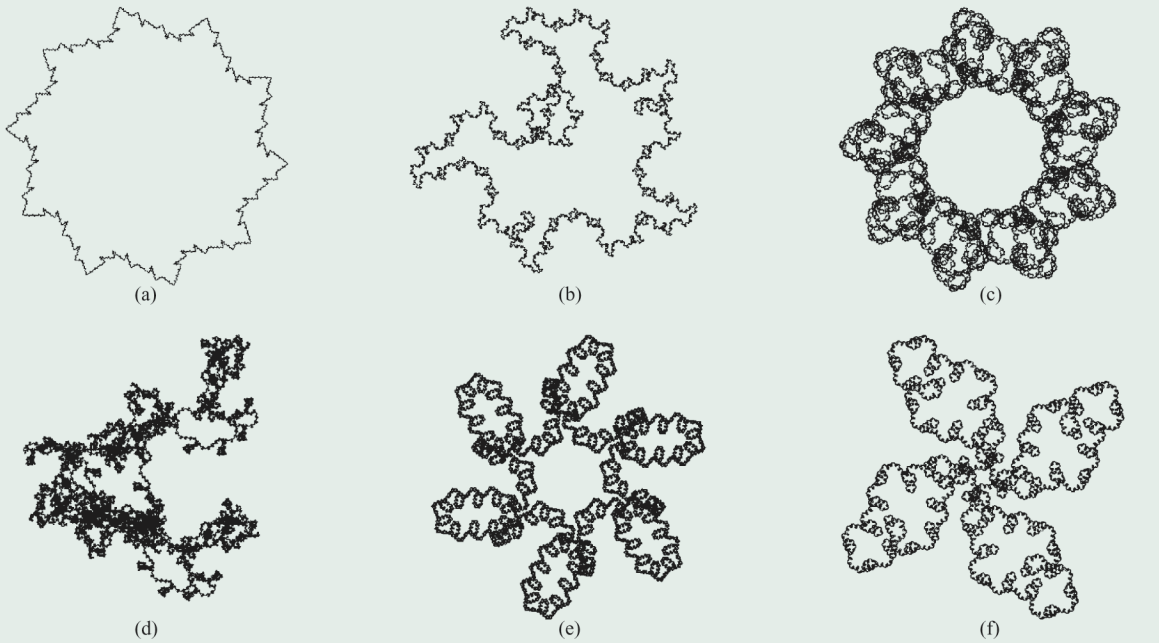

Figure	*Dimension*	*Angle* (degrees)	*Number of generations*	*Genotype size*	*L-system word developed*	*Axiom*
4(a)	1.26178...*	36	27	35	+F+F−−F−F+F−F+F+	F−F−F−F−F−F−F−F−F−F
4(b)	1.5018	45	2000	17	++F−F−−F+F+F+FF−−−F−F++	F−−−F++F−−−F−F−−F
4(c)	1.8998	40	1460	41	++F−FF−F−F−−−F−F−−F−−F−F++	F+F+F+F+F+F+F+F+F
4(d)	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
4(e)	1.7005	60	18	22	++FF+F+FFF+F+F+FFF+FFF+F+F−F+F−−−−	F−F−F−F−F−F
4(f)	1.6006	45	275	30	+F−F−−F−−F+FF−−F+F−F−FF+FF−F−−F	F++F++F++F

* *Same as Von Koch's snowflake.*

Figure 4

A few fractal curves evolved by our method.

Table 4 shows some of the results obtained with our second experiment, in which the turtle angle itself was subject to evolution by means of the genetic algorithm.

Conclusions and future research lines

Grammatical evolution has been applied to generate and evolve Lindenmayer grammars that represent fractal curves with a predefined fractal dimension. The procedure we have 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 in which 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 toward 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 predefined 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 to 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, when compared with 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:

- 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.

Table 4 A set of tests in which the turtle angle was also evolved.

Target dimension	*Actual dimension*	*Number of generations*	*Genotype size*	*Angle evolved* (degrees)
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

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 and 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 Visual Complexity of Pollock's Dripped Fractals," *Proceedings of the International Conference of Complex Systems*, 2002; see also *http://materialscience.uoregon.edu/taylor/art/Taylor1CCS2002.pdf*.
4. P. M. Iannaccone and M. Khokha, Eds., *Fractal Geometry in Biological Systems: An Analytical Approach*, CRC Press, Boca Raton, FL, 1996.
5. W. Bauer and C. D. Mackenzie, "Cancer Detection via Determination of Fractal Cell Dimension," presented at the Workshop on Computational and Theoretical Biology, Michigan State University, April 24, 1999; see also *http://www.pa.msu.edu/~/bauer/cancer/cancer.pdf*.
6. K. Danzer, J. F. van Staden, and D. T. Burns, "Concepts and Applications of the Term 'Dimensionality' in Analytical Chemistry," *Pure Appl. Chem.* **74,** No. 8, 1479–1487 (2002).
7. F. Torrens, "Fractals Hybrid Orbitals in Protein Models," *Complexity International* **8** (2001); see *http://www.csu.edu.au/ci/val108/torren01/*.
8. S. Mondoloni and D. Liang, "Airspace Fractal Dimension and Applications," presented at the Third 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 & Opt. Technol. Lett.* **29,** No. 4, 215–219 (2001).
10. A. Lindenmayer, "Mathematical Models for Cellular Interactions in Development" (in two parts), *J. Theor. Biol.* **18,** 280–315 (1968).
11. G. Herman and G. Rozenberg, *Developmental Systems and Languages*, North-Holland, Amsterdam, 1975.
12. E. G. Giessmann, "Generation of Fractal Curves by Generalization of Lindenmayer's L Systems," *Proceedings of the First IFIP Conference on Fractals in the Fundamental and Applied Sciences*, H. O. Peitgen, J. M. Henriques, and L. F. Penedo, Eds., North-Holland, Amsterdam, 1991, pp. 147–157.
13. P. Prusinkiewicz, "Graphical Applications of L-Systems," *Proceedings of Graphics Interface 86 and Vision Interface 86*, M. Wein and E. M. Kidd, Eds., Vancouver, BC, 1986, pp. 247–253.
14. M. Alfonseca and A. Ortega, "A Study of the Representation of Fractal Curves by L Systems and Their Equivalences," *IBM J. Res. & Dev.* **41,** No. 6, 727–736 (1997).
15. M. Alfonseca and A. Ortega, "Representation of Some Cellular Automata by Means of Equivalent L Systems," *Complexity International* **7,** 1–16 (2000), ISSN: 1320-0682; see *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," *Comput. & Graph.* **17,** No. 1, 85–88 (1993).
17. G. Rozenberg and A. Salomaa, Eds., *Lindenmayer Systems. Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, Springer-Verlag, Berlin, 1992.
18. M. Alfonseca 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. Alfonseca and A. Ortega, "Determination of Fractal Dimensions from Equivalent L Systems," *IBM J. Res. & Dev.* **45,** No. 6, 797–805 (2001).

20. B. Guiheneuf, S. Jaffard, and J. L. Vehel, "Two Results Concerning Chirps and 2-Microlocal Exponents Prescription," *Appl. & Computational Harmonic Anal.* **5,** 487–492 (1998).
21. K. Daoudi, J. L. Vehel, and Y. Meyer, "Construction of Continuous Functions with Prescribed Local Regularity," *J. Constructive Approximation* **14,** No. 3, 349–385 (1998).
22. S. Jaffard, "Functions with Prescribed Holder Exponent," *Appl. & Computational Harmonic Anal.* **2,** No. 4, 400–401 (1995).
23. M. O'Neill and C. Ryan, "Grammatical Evolution," *IEEE Trans. Evolutionary Computation* **5,** No. 4, 349–358 (2001).
24. M. O'Neill and C. Ryan, "Evolving Multi-Line Compilable C Programs," *Proceedings of the Second European Workshop on Genetic Programming* (*EuroGP'99*), 1999; lecture notes in *Computer Science* **1598,** 83–92 (1999).
25. M. O'Neill and C. Ryan, "Under the Hood of Grammatical Evolution," *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO'99*), W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., Morgan Kaufmann Publishers, San Mateo, CA, 1999, pp. 1143–1148.
26. M. O'Neill and C. Ryan, "Genetic Code Degeneracy: Implications for Grammatical Evolution and Beyond," *Proceedings of the Fifth European Conference on Artificial Life* (*ECAL'99*), 1999, pp. 149–153.
27. C. Ryan, J. J. Collins, and M. O'Neill, "Grammatical Evolution: Evolving Programs for an Arbitrary Language," *Proceedings of the First European Workshop on Genetic Programming* (*EuroGP'98*), 1998; lecture notes in *Computer Science* **1391,** 83–95 (1998).
28. C. Ryan and M. O'Neill, "Grammatical Evolution: A Steady State Approach," *Proceedings of the Joint Conference on Information Sciences*, 1998, pp. 419–423.
29. C. Ryan and M. O'Neill, "Grammatical Evolution: A Steady State Approach," *Proceedings of the Third Annual Genetic Programming Conference* (*GP'98*), 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, Eds., Morgan Kaufmann Publishers, San Francisco, 1998, pp. 180–185.
30. C. Ryan, M. O'Neill, and J. J. Collins, "Grammatical Evolution: Solving Trigonometric Identities," *Proceedings of the Fourth International Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, and Rough Sets* (*Mendel'98*), 1998, pp. 111–119.
31. M. Kimura, *The Neutral Theory of Molecular Evolution*, Cambridge University Press, Oxford, England, 1983.
32. G. Ochoa, "On Genetic Algorithms and Lindenmayer Systems," *Proceedings of Parallel Problem Solving from Nature, International Conference on Evolutionary Computation* (*PPSN IV*), 1998; 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 (Jordan B. Pollack, Advisor) in partial fulfillment of the requirements for the Ph.D. degree, 2003.
34. C. Jacob, "Genetic L-System Programming," *Proceedings of Parallel Problem Solving from Nature, the International Conference on Evolutionary Computation* (*PPSN III*), 1994, Springer-Verlag, Berlin; lecture notes in *Computer Science* **866,** 334–343 (1994).
35. C. Traxler and 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*, N. Magnenat-Thalmann and V. Skala, Eds., University of West Bohemia, Plzen, Czech Republic, 1996.
36. A. Ortega, M. de la Cruz, and M. Alfonseca, "Parametric 2-Dimensional L Systems and Recursive Fractal Images: Mandelbrot Set, Julia Sets and Biomorphs," *Comput. & Graph.* **26,** No. 1, 143–149 (2002).

Received January 14, 2003; accepted for publication March 3, 2003

Alfonso Ortega *Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain (Alfonso.Ortega@ii.uam.es).* Dr. Ortega is currently a lecturer at the University. He formerly lectured at the Universidad Pontificia de Salamanca and worked at LAB2000 (an IBM subsidiary) as a software developer. He holds a doctorate in computer science from the Universidad Autónoma. Dr. Ortega has published 14 technical papers on computer languages, complex systems, graphics, and theoretical computer science, and has collaborated in the development of several software products.

Abdellatif Abu Dalhoum *Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain (Abdel.Latif@ii.uam.es).* Mr. Dalhoum received his B.Sc. degree in computer science from Al-Mousel University in Iraq, and his M.Sc. degree in computer engineering from the Universidad Autónoma de Madrid, Spain. Currently, he is doing Ph.D. research at the Universidad Autónoma. Mr. Dalhoum is particularly interested in genetic algorithms, Lindenmayer systems, and cellular automata.

Manuel Alfonseca *Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain (Manuel.Alfonseca@ii.uam.es).* Dr. Alfonseca is a professor at the University. He was formerly a Senior Technical Staff Member at IBM, having worked from 1972 to 1994 at the IBM Scientific Center in Madrid. Dr. Alfonseca was one of the developers of the APL/PC interpreter and related products; he has worked on computer languages, simulation, complex systems, graphics, artificial intelligence, object orientation, and theoretical computer science, and has published several books and about 180 technical papers, as well as 60 papers on popular science in a major Spanish newspaper. He is an award-winning author of 21 published books for children. Dr. Alfonseca holds a doctorate in electronics and an M.Sc. degree in computer science from the Universidad Politécnica de Madrid. He is an emeritus member of the IBM Technical Expert Council and a member of the Society for Computer Simulation (SCS), the New York Academy of Sciences, the IEEE Computer Society, the ACM, the British APL Association, and the Spanish Association of Scientific Journalism.