Generating grammatical plant models with genetic algorithms

Luis E. Da Costa, Jacques-André Landry
Laboratoire d'Imagerie, Vision, et Intelligence Artificielle (LIVIA)
École de Technologie Supérieure, Montréal, Québec (Canada)
E-mail: costa@livia.etsmtl.ca, jacques.landry@etsmtl.ca

Abstract

A method for synthesizing grammatical models of natural plants is presented. It is an attempt at solving the inverse problem of generating the model that best describes a plant growth process, presented in a set of 2D pictures. A geometric study is undertaken before translating it into grammatical meaning; a genetic algorithm, coupled with a deterministic rule generation algorithm, is then applied for navigating through the space of possible solutions. Preliminary results together with a detailed description of the method are presented.

1 Background and motivation

The detailed study of a set of plants from an agricultural field or a forest is a precious source of information about their health, the treatments that the plants have undergone and, consequently, about the appropriate management strategies. However, there is a physical impossibility in bringing to the field the specialized equipment needed to perform such a study. An approach to solve this constraint is to build a detailed model in order to obtain a-priory knowledge about the plants we wish to observe or to perform virtual computer studies. Our first attempt is to model dichotomous trees such as the great Maple.

Based on the self-similarity observed in its development (Mandelbrot [6], Barnsley [1]), a plant can be thought of as a natural ("*existing in nature*") representation of a fractal; therefore, it is modeled using tools created by the fractal research community. We are specially interested in a family of models called **L-S**ystems (for **L**indenmayer **S**ystems, [5]). With the addition of simple geometric features, **L-S**ystems have been used extensively for visualization of natural developmental processes (see, for example, [7] and [8])[1].

However, *observing* a natural growth process and then *generating* a model that describes it is still more of an art than a structured approach more suitable to science. In section 2 we present this *inverse problem*; a solution is proposed in 3. Preliminary results are discussed in 4, and we conclude in 5.

[1] In this paper, we won't go into any deeper details concerning the basic concept and definition of **L-S**ystems. These can be found in [9]

2 Evolving a model to describe plant growth: the inverse problem

2.1 The inverse problem for a fractal

As mentioned, a tree can be represented using its fractal characteristics. To represent a given tree, one must search for a grammar that generates a fractal that best represents the sought tree. The question to solve is then: how to find a model to generate a *specific* fractal figure?

Answers from different disciplines exist to related questions: in architecture, Coates *et al* ([2], chap. 14) combined **L-S**ystems with genetic programming for exploring designs. In [10], Lindenmayer and Prusinkiewicz generated **FASS** (*space-***F***illing, self-***A***voiding,* ***S****imple and self-****S****imilar*) curves with graph rewriting techniques. In [11], Vanyi *et al.* developed a system for graphically describing the circulation of blood in a human retine, coupling evolutionary methods coupled with a restricted form of **L-S**ystems. And, finally, the *Collage Theorem* establishes a way of solving the inverse problem using *Iterated Function Systems* (**IFS**) ([1]).

2.2 The inverse problem for plant growth

The *growth process* of a plant can be described as a suite of fractal figures, each one representing a specific stage in time of this process; a *model* for this suite then describes the growth process of this plant. The question to answer at this stage is then: having *observed* a growth process of a plant (or tree), what is the model that best describes it?

The use of **L-S**ystems (see [5, 7, 8, 9] for details) in an attempt to answer this question is quite appropriate as the recursive nature of the grammar usage makes them generate intermediate results in discrete time steps, much like observing a plant growth in predefined, equally spaced time intervals. Prusinkiewicz and Hanan, in [9], extensively demonstrate the usefulness of **L-S**ystems for answering this question with what is known as *computational botany*. Somehow guided by a different objective,

Koza, in [4], uses genetic programming for generating the (one and only) rule of an **L-S**ystem.

The question that we address in this research project goes still a step further, adding a geometrical component: the object whose growth process is studied is three-dimensional, but the temporal information that we have is two-dimensional (a suite of 2D pictures). With this new restriction in mind, we propose in section 3 a method for generating a growth model for a tree (expressed as an **L-S**ystem grammar), having a set of 2D pictures. No hard restriction is set on the *shape* of the pictures (they don't have to be taken from any specific point in space) nor on their *age* (*when* they were taken is not important, as long as we have this information).

3 Proposal

This research problem, as stated, implies working with 2 parallel views of trees: the *geometric* representation, that defines how the tree is formed as a composition of shapes, and the *syntactic* representation: the **L-S**ystem string that is interpreted as the named geometric figure. We have to be able to handle both views at the same time, as the objective is to generate the same geometry, but this can only be done through the correct generation of a syntactic model.

In this section, a characterization of the kind of solutions we are looking for is described (3.1), and how this is represented in the syntactic domain is inferred (3.2).

3.1 Geometric considerations

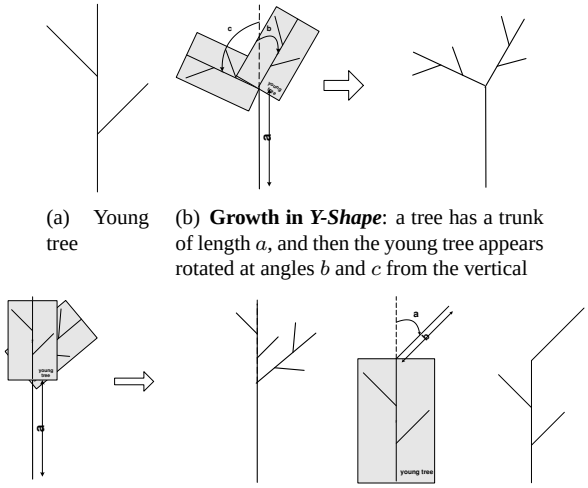
Our goal being to model natural trees, the geometric nature of tree growth should be reflected in the chosen **L-S**ystem rules. We defined 3 ways in which a tree grows, exemplified with a series of drawings. If the shape of a given *young tree* is as in Fig. 1(a), the *natural* growth is a combination of **Y-*shape*** (Fig. 1(b)), **T-*shape*** (Fig. 1(c)) and **B-*shape*** (Fig. 1(d))

3.2 Grammatical translation

As shown in Fig. 1(b,c,d), the different types of growth amount to different ways of combining the "young tree" while forming a larger tree. We have defined 3 syntactic operators, named **B**, **T**, and **Y**, translating the geometric relations explained in Fig. 1 to **L-S**ystem notation. These are called the *growth* operators:

1. **Fig. 1(d)**, representing **B**, would be $S + (a)F(b)$. Extending it to a three dimensional space and specifying the contexts, the complete **L-S**ystem rule would be:

   left ctx. $< B(a, b, c) >$ right ctx. $\rightarrow S{+}(a)\&(c)F(b)$



(a) Young tree

(b) **Growth in *Y-Shape***: a tree has a trunk of length a, and then the young tree appears rotated at angles b and c from the vertical

(c) **Growth in *T-Shape***: a tree has a trunk of length a, and then the young tree appears rotated an angle b from the vertical

(d) **Growth in *B-Shape***: from the young tree grows a line of length b rotated an angle a from the vertical

Fig. 1. Growth of a tree

   This notation presents the advantage of having non-fixed contexts, allowing for a family of rules of this shape. Following this idea, we also derive a set of rules with the geometry of Fig. 1(b) and (c) (in three dimensions):

2. **Fig. 1(b)**:

   $$Y(a, b, c, d) \rightarrow F(a)[+(b)\&(d)S][-(c)\&(d)S]$$

3. **Fig. 1(c)**:

   $$T(a, b, c) \rightarrow F(a)[+(b)S]S$$

Now that we have defined the operators, we can take a closer look at the *primordial* components of the tree (Fig. 1(a)); this "young tree" has a specific meaning as part of an **L-S**ystem: it is the *axiom*, $\mathcal{S}$. As mentioned, the operators only use combinations of this first tree for building the whole tree; so it is important to choose a large enough *family of axioms* for representing a large number of different type of trees, if we want our method to be general.

We have chosen this family to have the following syntactic form: $P_{ts}GP_{ts}GP_{ts}$, where:

- G is one of B, T, or Y (correspondingly parameterized) and
- P_{ts} is the **L-S**ystem string of a *primordial tree shape*.

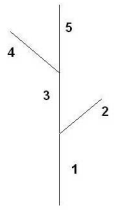


Fig. 2. General primordial tree shape

This primordial shape is a generalization of the shape shown in Fig. 2. It is represented by the concatenation of 5 parameterized **L-S**ystem regular expressions: part 1, 3, and 5 in Fig. 2 are $F(l)^{0/1}$ (*"straight line of length l"*), and part 2 and 4 are $[\pm(\alpha_1)\&(\alpha_2)F(l)]^{0/1}$ (*"straight line of length l, growing from the vertical trunk at a 3D angle specified by α_1 and α_2"*). The exponent $0/1$ translates the possibility of presence/absence of the term.

3.3 The L-System family of solutions

These *translations* generate a *parametric context-sensitive* **L-S**ystem family (Table 1, with rules representing objects in 3D)[2].

► 12 *reserved symbols*: $\mathcal{R_S}$ = {F,f,+,-,∧,&,\,/,(,),[,]}
► *Alphabet*: $\mathcal{T} = \mathcal{R_S} \cup \mathbb{R}$
► A *parameterized Axiom* $\mathcal{S} \in T^+$
$\mathbf{lc_1^b} <$ B(a,b,c) $> \mathbf{rc_1^b} \rightarrow$ S+(a)&(c)F(b)
⋮
$\mathbf{lc_n^b} <$ B(a,b,c) $> \mathbf{rc_n^b} \rightarrow$ S+(a)&(c)F(b)
$\mathbf{lc_1^y} <$ Y(a,b,c,d) $> \mathbf{rc_1^y} \rightarrow$ F(a)[+(b)&(d)S][-(c)&(d)S]
⋮
$\mathbf{lc_m^y} <$ Y(a,b,c,d) $> \mathbf{rc_m^y} \rightarrow$ F(a)[+(b)&(d)S][-(c)&(d)S]
$\mathbf{lc_1^t} <$ T(a,b,c) $> \mathbf{rc_1^t} \rightarrow$ F(a)[+(b)&(c)S]S
⋮
$\mathbf{lc_k^t} <$ T(a,b,c) $> \mathbf{rc_k^t} \rightarrow$ F(a)[+(b)&(c)S]S

Table 1. The shape of the family of grammars

3.4 Method: generating a grammar

Now the problem is more specific:

> *Having a list of figures $\mathcal{L}_t$, find a grammar of the family defined in Table 1 whose graphical interpretation best approaches $\mathcal{L}_t$*

Our approach for solving this is shown in Alg. 1

[2] lc_i^r stands for *left context i for rule type r* and rc_j^r for *right context j for rule type r*

Algorithm 1 Grammar synthesis

1: **procedure** GENERATEGRAMMAR($\mathcal{L}_t$: target figures; $\mathcal{P}_t$: spatial and temporal information about $\mathcal{L}_t$)
2: **repeat**
3: Choose a parameterized grammar M_a from the *axiom space* ▷
4: Generate the best set of *production rules* (P_R) for the current axiom ▷
5: $G = M_a \oplus P_R$ ▷ current grammar
6: **until** G is *good enough*
7: Return G
8: **end procedure**

The exploration of the axioms' space (line 1.3) is implemented as a genetic algorithm, described in 3.4.1. The production rules generation (line 1.4) is described in (3.4.2), and the fitness function for the whole process is presented in 3.4.3.

3.4.1 Choosing axioms: Each symbol of the axiom (as described in 3.2 and Table 1) is encoded as a gene in the chromosome, which is then 64 genes long. Straightforward 1-point crossover and mutation operators are implemented on such representation.

3.4.2 Navigating the production rules space: In this approach, we derive only the rules that could possibly expand a specific axiom. We use information from Alg.1: generation of the rules comes *after* choosing a possible axiom. The general method calculates each possible outcome of a string, starting from the axiom, and then derives the rule that would generate this string. For example, if the axiom is $\mathcal{S} : B(1,2,3)F(4)$, then its first derivation will be

$$\mathcal{S}{+}(1)\&(3)F(2)F(4) \qquad \textbf{or} \qquad B(1,2,3)F(4)$$

depending on whether there is a rule in the grammar G that replaces $B(1,2,3)$ or not. So G will be either G_1 or G_2:

$$G_1 = \begin{cases} \text{Axiom} = & \mathcal{S} : B(1,2,3)F(4) \\ \mathbf{B(a,b,c)} > F \rightarrow & \mathcal{S} + (a)\&(c)F(b) \end{cases}$$

$$G_2 = \begin{cases} \text{Axiom} = & \mathcal{S} : B(1,2,3)F(4) \\ & \emptyset \end{cases}$$

Each possibility defines a current work string, that in turn could be replaced by the rules of a grammar; this iterative procedure is repeated for the number of generations indicated in the information about the target figures.

In that way, starting from the axiom, we build a graph in which each node is the state of the generation of a string, and whose leading transition is the rule that was applied in order to obtain the string (see Fig. 3). This type of graph does not have cycles, so it is a tree; each path from the root to each one of the leaves is a complete grammar, and a possible solution to the optimization problem. Finding the *best grammar* is now equivalent to *evaluating each grammar path with an appropriate fitness function (3.4.3) and choosing the one with better fitness*.

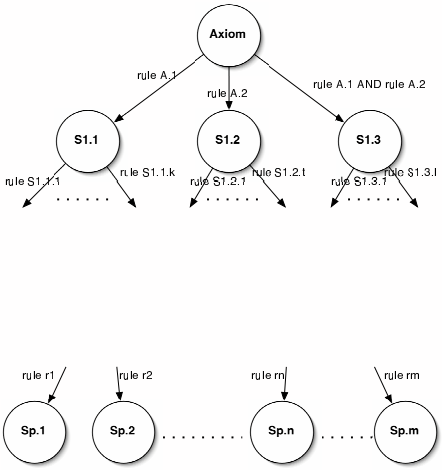


Fig. 3. Grammar evaluation graph

3.4.3 Fitness function: The fitness of a grammar G is a *distance* between a list of target figures $\mathcal{L}_t$ and the figures generated by G. The *distance* between 2 figures is evaluated by a comparison between their respective fractal dimensions (F, calculated using the box-counting algorithm) and their size (S):

$$\mathcal{D}(f_1, f_2) = \alpha*(F(f_1)-F(f_2))+(1-\alpha)*(S(f_1)-S(f_2))$$

α is a weighting estimator of the importance of each measure; for this paper, we fixed $\alpha = 0.5$

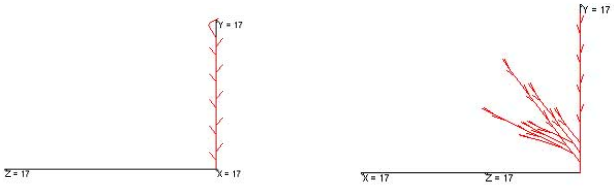
The fitness of a grammar G ($\mathcal{F}(G)$) is the sum of all the distances $\mathcal{D}$

4 Results and discussion

We implemented an environment to experiment with the visual appearance and with the formal definitions of **L-S**ystems ([3]). We then built a synthetic database of 2D pictures of trees generated using 3D grammars. The proposed method is being evaluated for its ability to generate grammars that can evolve trees corresponding to the source synthetic pictures.

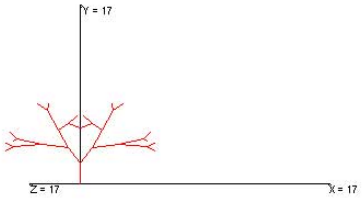
For this first set of tests we aimed at finding perfect fitness (fitness = 0, because *fitness* as in 3.4.3 is a sum of distances). Fitness being the ending condition in Alg. 1, this severe rule, associated with the cardinality of the axiom space being $\sim 10^{10}$, meant that the exploration of the whole space was unthinkable, even with very powerful machines. However, such a drastic constraint (fitness =0) was useful in evaluating the proposed approach when dealing with simple geometric figures and a subset of the grammar. Not only did this allowed to verify if we generated similar trees, but also similar grammars (remember that we are using synthetic trees).

To this end, we studied the geometry of grammars with simple axioms and one growth rule, either B, T, or Y. Under certain conditions, the geometry of each class of grammars is very distinctive, and is presented (in its 8th. iteration) in Fig. 4. We gave as inputs to our method



(a) Grammar with axiom: F[+F]F[-F]FB(30,1,45) and the B growth operator as the sole rule

(b) Grammar with axiom: T(1,30,45)F[+F]F[-F]F and the T growth operator as the sole rule



(c) Grammar with axiom: FY(1,45,45,45) and the Y growth operator as the sole rule

Fig. 4. Input to our experiences

the sequences generated using each of the growth rule, and hoped to obtain not only trees that were similar but also the same grammar as entered. We repeated the experiences 10 times with each sequence, and present here some of our observations:

1. for all replicates, the resulting grammar generated a tree that was similar to the input picture
2. in 8 replicates out of 10, the algorithm did not find the exact grammar as entered, and the search continued even though the solutions were very close and the fitness near 0.

3. in 2 replicates out of 10, the stochastic process quickly generated a perfect solution.

4. the solutions were generated in reasonable amounts of time. However, once again, the ending condition requiring a perfect fit (fitness = 0) did not allow the algorithm to stop when acceptable solutions were encountered.

5 Conclusion and future work

These first results are very promising, and demonstrate that the proposed approach is a solution to the inverse problem. We now have to improve the intuitiveness of our fitness function: make it closer to our expectations when comparing 2 natural trees. We, as humans, do not expect a tree to look *exactly* the same in 2 different pictures. However, we can recognize different pictures as coming from the same tree. To this end, additional criteria, such as comparing the number of segments, the compactness, or even the texture of the trees, would definitely improve the value returned by the fitness function.

Another important consideration is in devising a more objective way of evaluating our method. This is actually quite difficult to achieve, as we are evaluating the results as humans, in a subjective manner. However, we think that we would greatly benefit from a form of "mathematical evaluation" of our results.

We believe that the proposed approach for generating grammatical models from example is novel and very promising, even though the presented results are still preliminary. The next step will be to generate a consistent set of results, along with extending it to more complicated grammars (for example, stochastic). Finally, we would like to draw some conclusions about the correctness, usefulness and robustness of the proposed method.

References

[1] Michael F. Barnsley. *Fractals everywhere*. Academic Press Limited, London, 2 edition, 1993.

[2] P J Bentley. *Evolutionary Design By Computers*. Morgan Kaufmann, San Francisco, CA, 1999.

[3] Luis E Da Costa and Jacques-André Landry. A convivial visualization environment for lsystems. In *FRACTAL 2004: 8th International Multidisciplinary Conference*. Vancouver, BC, 2004.

[4] J.R. Koza. Discovery of rewrite rules in lindenmayer systems and state transition rules in cellular automata via genetic programming. In *Symposium on Pattern Formation (SPF-93)*. Claremont, California, 1993.

[5] A Lindenmayer. Mathematical models for cellular interaction in development. parts i and ii. *Journal of Theoretical Biology*, 18:280–299 and 300–315, 1968.

[6] B B Mandelbrot. *The fractal geometry of nature*. San Francisco, 1982.

[7] P Prusinkiewicz. Graphical applications of l-systems. In *Proceedings of Vision Interface '86*, pages 247–253. 1986.

[8] P Prusinkiewicz. Applications of l-systems to computer imagery. In H Ehrig, M Nagl, A Rosenfeld, and G Rozenberg, editors, *Graph grammars and their applications to computer science; Third international workshop*. Springer-Verlag, 1987.

[9] P Prusinkiewicz. Lindenmayer systems, fractals, and plants. *Lecture Notes in Biomathematics*, (79), 1989.

[10] P Prusinkiewicz and A Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[11] Robert Vanyi, Gabriella Kokai, Zoltan Toth, and Tiinde Reto. Grammatical retina description with enhanced methods. In *Proceedings of Genetic Programming 2000*. 2000.