Modeling Leaf Shapes Using L-systems and Genetic Algorithms

Yodthong Rodkaew
Intelligent System Lab (ISL),
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
43718130@student.chula.ac.th

Suchada Siripant
Advanced Virtual and Intelligent Computing Research Center (AVIC),
Department of Mathematics and Computer Science,
Faculty of Science, Chulalongkorn University
ssuchada@chula.ac.th

Chidchanok Lursinsap
Advanced Virtual and Intelligent Computing Research Center (AVIC),
Department of Mathematics and Computer Science,
Faculty of Science, Chulalongkorn University
lchidcha@chula.ac.th

Prabhas Chongstitvatana
Intelligent System Lab (ISL),
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
prabhas@chula.ac.th

Tadahiro Fujimoto
Department of Computer Science,
Faculty of Engineering, Iwate University
fujimoto@cis.iwate-u.ac.jp

Norishige Chiba
Department of Computer Science,
Faculty of Engineering, Iwate University
nchiba@cis.iwate-u.ac.jp

Abstract

This work presents a method that combines two techniques: L-systems and Genetic Algorithms (GA) to search for a rewriting expression describing leaf shapes. An L-system is used to construct the shape of a given rewriting expression and GA is used to search for the rewriting expression's fitting parameters. The replacement of real value parameters with tag-functions is introduced. The result shows that the proposed method produces an acceptable output.

Key words: CG, Natural Phenomena, Modeling, Leaf, L-systems, Genetic Algorithms

1. Introduction

In 1968, L-systems [1] were introduced by the biologist, Aristid Lindenmayer, to create a realistic plant form by a context-free rewriting expression with conditional and stochastic rule selection. The computer graphical output from computer software that uses L-systems [2,3] resembles a real plant. However, there are some plant parts that cannot be derived by a rewriting expression such as leaves or flowers. In the computer software [2,3], predefined leaf and flower shapes are used to compose a plant. The method described in [1, pp. 120-127] realizes predefined expressions for leaf edges.

This paper's interest lies in finding a rewriting expression for a leaf network (Fig.1). The method proposed in [4] tried to construct a primary branch network with a given expression, the leaf shape being modified by changing parameters. However, the modification of the parameters by humans is difficult because there are many parameters. This paper proposes a method to construct a primary branch and solve the problem of parameter fitting using Genetic Algorithms [5].

This paper is organized as follows: Section 2 and 3 briefly introduce L-systems and Genetic Algorithms. The experiment and the results are presented in Section 4. Finally, Section 5 contains the conclusion and comments on future work.

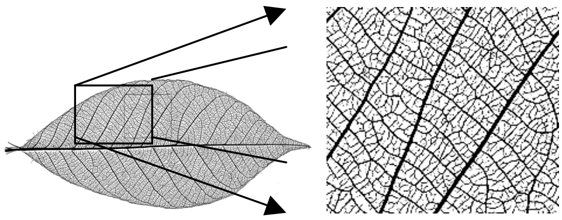


Figure 1: The leaf shape and leaf network.

2. L-systems

From [1], a parametric 0L-system, which is context-free, operates on *parametric words*, which is a string consisting of *letters* and *parameters*, called *modules*. The letter as alphabet is denoted by V, and the set of parameters is the set of the real number $\Re$. A module with letter $A \in V$ and parameters $a_1, a_2, \ldots, a_n \in \Re$ is denoted by $A(a_1, a_2, \ldots, a_n)$. Every module belongs to the set $M = V \times \Re^*$, where $\Re^*$ is the set of finite sequences of parameters. The set of all module strings is denoted by $M^* = (V \times \Re^*)^*$, and the set of all nonempty strings is denoted by $M^+ = (V \times \Re^*)^+$.

The real-valued *actual* parameters appearing in the words correspond with *formal* parameters, which may occur in the specification of L-system productions. Let Σ be a set of formal parameters. A parametric 0L-system is defined as an ordered quadruple G = (V, Σ, ω, P), where:

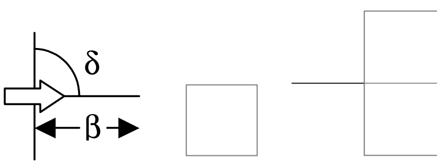
- V is the alphabet of the system.
- Σ is the set of formal parameters.
- $\omega \in (V \times \Re^*)^+$ is a nonempty parametric word called the axiom.
- $P \subset (V \times \Sigma^*)$ is a finite set of productions.

A logical expression and an arithmetic expression with a parameter from Σ can be included in the system.

One example is given as follows, when the alphabets V in the system are {F, R}, which may occur many times in a string. Each letter is associated with a rewriting rule. The rule F → FRF means that letter F is to be replaced by FRF. The rewriting process starts from a distinguished string called the *axiom* or ω. Given the axiom string F, in the first derivation step, the string F is replaced by string FRF to become string FRF. In the second derivation step, the string FRF is replaced by string FRFRFRF.

2.1 Drawing mechanism in L-systems

In L-systems, the drawing is based on *turtle graphics*. The *turtle* state is defined as a triplet (x, y, α), where the *Cartesian coordinates* (x, y) represent the turtle's position and α is the direction of the turtle. A *step size* β and an *angle increment* δ are given. In Fig.2, they are given with β = 5.0 and δ = 90.0°. The symbol F means "move forward" a step, symbol L means "turn left" by an angle increment, and symbol R means "turn right" by an angle increment. For example, the string FRFRFRF draws a rectangle (Fig. 2b). The symbols [and] mean a stack. Symbol [pushes the current (x, y, α) on the stack, while symbol] pops a (x, y, α) from the stack and the (x, y, α) is assigned to the current one (Fig. 2c).



(a) (b) (c)

Figure 2: (a) The turtle. (b) The picture obtained from string FRFRFRF. (c) The picture obtained from string F[LFRF][RFLF]F.

2.2 Parametric words

One or more parameters can be associated with a symbol. If symbol F means move forward, then F(5) means move forward by 5 pixels.

F(α) Move forward by α pixels
R(α) Turn right by α angle
L(α) Turn left by α angle

Users can define a new parametric rule. Using an *arithmetic expression* ε(Σ), the definition below has valid productions in the L-system:

ω : A → B(1)
P_1: B(a) → C(a, a+1)
P_2: C(a, b) → B(a)C(b, a+b)

The result after the 1st derivation is: C(1,2). The result after the 2nd derivation is: B(1)C(2,3). The result after the 3rd derivation is: C(1,2)B(2)C(3,5). The result after the 4th derivation is: B(1)C(2,3)C(2,3)B(3)C(5,8).

2.3 Tag-functions

The parametric words need derivation steps; symbols and parameters change by the L-system production rules. However, it is considered to be difficult to construct suitable production rules for generating the leaf shape similar to a real leaf using purely the expression style described in Section 2.2. Therefore, in our method, *tag-functions* are introduced to replace parameters. A tag function can be given any suitable function. In addition, the tag-function can reduce computational load because it replaces repeated derivations by memorizing them in tables; hence, it reuses the calculation of the parameter values. Users can change a function value without requiring the regeneration of an axiom derivation.

The tag-functions used in this paper are described in Table 1. In a string, a tag-function begins with symbol < and ends with >, such as <S> (see Section 2.4). After derivation, each tag function is assigned a number by the parser. This number is the index in the function table. When interpreting a tag-function, the turtle uses the derived axiom and looks up the value from the associated table to obtain a real value (Fig.3). The arguments *n* and *m* of the tag-functions in Table 1 vary from 0 to 1. They are added by the parser. The number *n* is increased by the derivation number, while the number *m* is increased by the number of the appearance of a tag-function symbol, such as <G>, in the derived string in one derivation stage.

The terminals L, R, and F mean turn left by δ, turn right by δ, and move forward by β, respectively, as described in Section 2.1. The constants δ and β are given by users. The symbol γ in Table 1 denotes a scaled value of the step size β. This means that the symbol F instructs the turtle to move forward by step size ($\gamma \times \beta$) pixels. The tag-functions D and E are defined by linear interpolation of the functions L_1, L_2, and L_3. The terminal ! signifies the reset of a state.

In our method, it is so important to determine the 7 functions S, P, Q, G, L_1, L_2, and L_3 so as to generate a good leaf shape. In the following, we use the term "tag-function" for these 7 functions for convenience, although L_1, L_2, and L_3 are not actually tag-functions. We will describe the method to obtain suitable functions for them in Section 3.

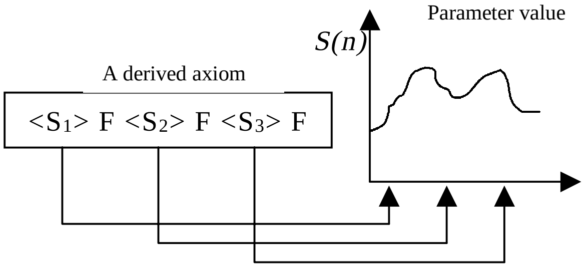


Figure 3: Conversion of tag-function to parameter value.

Table 1: Tag-functions.

Tag - Function	Meanings	Output Range
<S_n>	$\gamma = S(\mathrm{n})$	0.0 – 1.0
<P_n>	Turn left by P (n) angle	0.0 – 90.0
<Q_n>	Turn right by Q (n) angle	0.0 – 90.0
<G_m>	$\gamma = G(m)$	0.0 – 1.0
<$D_{m,n}$>	$\sigma = \lVert m \rVert$ $\theta = \begin{cases} (1-2\sigma)L_1(n) + 2\sigma L_2(n) \\ \quad where 0 \le \sigma \le 0.5 \\ 2(1-\sigma)L_2(n) + (2\sigma - 1)L_3(n) \\ \quad where 0.5 < \sigma \le 1 \end{cases}$ Turn left by θ angle	0.0 – 90.0
<$E_{m,n}$>	Same as <$D_{m,n}$>, but turn right by θ angle	0.0 – 90.0

2.4 Experiment's rule definition

In the experiment of this paper, the following grammar is used to produce the leaf shape:

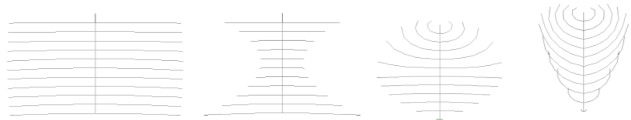
ω : A → LM!N
P_1: M → [BBBBBBBBBBB]
P_2: N → [CCCCCCCCCCC]
P_3: B → F[H]
P_4: C → F[I]
P_5: H → L<P><G>J
P_6: I → R<Q><G>K
P_7: J → <E><S><S>J
P_8: K → <D><S><S>K

The derivation proceeds up to the 8^{th} order as follows.

init : LM!N
1^{st} : L[BB…B]![CC…C]
2^{nd} : L[F[H]F[H]…F[H]]![F[I]F[I]…F[I]]
3^{rd} : L[F[L<P_3><G_1>J]F[L<P_3><G_2>J]...F[L<P_3><G_{11}>J]]!
[F[R<Q_3><G_1>K]F[R<Q_3><G_2>K]…F[R<Q_3><G_{11}>K]]
4^{th} : L[F[L<P_3><G_1><E><S_4><S_4>J]……]!
[F[R<Q_3><G_1><D><S_4><S_4>K]……]
5^{th} : L[F[L<P_3><G_1><E><S_4><S_4><E><S_5><S_5>J]……]!
[F[R<Q_3><G_1><D><S_4><S_4><D><S_5><S_5>K]……]
…
8^{th} : …

The subscript numbers of the tag-functions above indicate *n* or *m*. They are actually normalized to [0, 1] when they are given to the tag-functions as arguments. For example, <S_5> means that the function value S(5/8) is used, where 8 is the maximum derivation number. In this experiment, the step size β = 15.0 and the angle increment δ = 90.0°.

The above rule is based on the idea of deforming a skeleton (Fig.4a). The skeleton varies in shape by changing the parameters (Fig.4b-d).



(a) (b) (c) (d)

Figure 4: Skeleton shapes created by rules. (a) Original skeleton. (b) (c) (d) Various skeleton shapes deformed by adjusting function.

3. Genetic Algorithms

Genetic Algorithms (GA) [5] are based on an inspiration from natural selection. GA was developed by John Holland at the University of Michigan. It is a *robust* algorithm used for the search and optimization of solutions. Each solution is called an *individual*. A group of individuals is called a *population*. GA evaluates each individual to measure *fitness*. Some individuals who are in peak fitness condition are selected to produce *offspring* for the next *generation*.

In the first generation, GA randomly generates a population with specific parameters: the number of individuals, the length of an individual etc., and evaluates them. In the next generation, GA randomly selects some individuals with probabilities according to their fitness to produce offspring and modifies them by *genetic operators* (*reproduction*, *crossover*, and *mutation*). This process is repeated until the terminating condition is satisfied, such as an individual who can solve the problem is obtained or the specific number of generation is reached.

3.1 Individuals

In our method, GA is used for determining the 7 tag-functions in Table 1. We define the tag-functions as *β-Spline* functions [6], each of which is defined by 4 control points. For applying GA, an individual is given the information of a set of 7 tag-functions; concretely, it consists of 28 floating-point values, where there are 4 values a tag-function, as shown in Fig. 5.

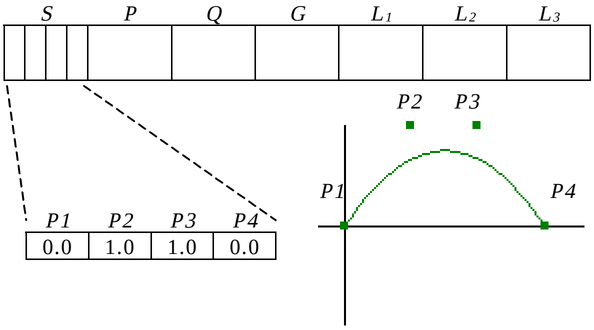


Figure 5: Tag-function represented by β-Spline.

3.2 Genetic Operators

The genetic operators used in this experiment are reproduction, crossover, and mutation. The reproduction is the duplication from parent to children. In the crossover, two individuals are selected as parents, a random point is selected to split each individual into two parts then these two parts are exchanged and recombined. This operator generates two offspring. The mutation operator produces an offspring by randomly changing four values in a parent.

3.3 Fitness Function

The fitness of an individual is evaluated by the fitness function that measures the distance between the output from the L-system and the target picture of a real leaf. In this experiment, only the outline of a leaf was considered. The fitness function is as follows:

$$\text{Fitness} = \sum_{i}^{n} (\mathbf{x}_i^t - \mathbf{x}_i^o)^2$$

where $\mathbf{x}$ is the coordinate (x, y), $\mathbf{x}_i^t$ is the outline of the target and $\mathbf{x}_i^o$ is the outline of the output from the L-system.

4. The Experiment

In our method, for each individual in the GA, the leaf shape is generated by the L-system rules using the 7 tag-functions defined by 28 floating-point values the individual has. By evaluating the fitness of the leaf shape using the fitness function, the individuals who produce good shapes are selected. After enough iteration of the GA, a set of 7 tag-functions that produce the shape similar to the target leaf shape is obtained.

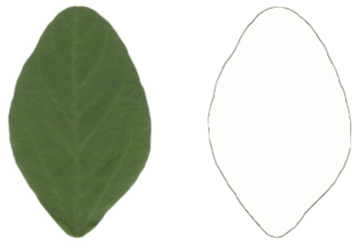
4.1 Genetic Parameters

The genetic parameters are as follows:

Number of Individual	200
Reproduction	20%
Crossover	40%
Mutation	40%
Number of Generation	200

4.2 Target

The target picture is taken from the outline of a real soybean leaf. A sample of the target is shown in Fig.6.



(a) (b)

Figure 6: (a) A soybean leaf and (b) its outline.

4.3 Results

The quality of the output from our system is satisfactory in terms of the similarity of the outlines. The leaf shape produced by the L-system with 7 tag-functions gradually becomes better and approaches the target shape as the generation of the GA increases. Figure 7 shows the graphs of the tag-functions and the produced leaf shape by the fittest individual of the first generation. On the other hand, Figure 8 shows the result of the final generation. The output matches closely to the target. This result is better than the shape produced by human adjusting of the parameters manually. Figures 9 and 10 show another example. Figure 11 shows CG images generated by applying texture mapping for the obtained shape of vein.

5. Conclusion

This paper has proposed a method that combines two techniques, L-systems and Genetic Algorithms, to search for a rewriting expression describing a leaf shape. The replacement of real value parameters by tag-functions has been introduced. The leaf shape from a rewriting expression looked satisfactory in terms of similarity to the outline of a target leaf. This result can be improved by modification of a rewriting expression. Currently, the fitness function evaluates only the outline of a leaf. For future work, in addition to the outline, we will attempt to evaluate the shape of vein to enhance the reality of the vein network. In addition, we will apply our method to create various kinds of leaves, such as maple and oak. From a viewpoint of photo-realistic CG, it is important to develop a method for adding natural fractuation to the shape of vein. Moreover, it is interesting to simulate the opening and growing process of a leaf bud using the geometrical theory of folding like that studied by K.Kaino et al.[7,8].

References

[1] Prusinkiewicz P. and Lindenmayer A., *The Algorithmic Beauty of Plants.*, Springer-Verlag, 1996.

[2] Cynthia F. Kurtz and Paul D. Fernhout, *PlantStudio.*,

Computer Software, http://www.kurtz-fernhout.com/PlantStudio/.
[3] Chuai-aree S., *PlantVR.*, Computer Software, personal communication.
[4] Lursinsap C., Sophatsathit P., and Siripant S., *Simulation of Leaf Growth Based On A Rewriting System: A Unified Leaf Model.*, Advanced Virtual and Intelligence Computing (AVIC) Research Center, Chulalongkorn University., Technical Report No. 01.01.2000.
[5] Goldberg D.E., *Genetic Algorithms in Search, Optimization, and Machine learning.*, Addison-Wesley, Redwood City, 1989.
[6] Watt A., Watt M., *Advanced Animation and Rendering Techniques.*, Addison-Wesley, New York, pp.90-91.
[7] K.Kaino, Geometry of Folded Pattern of Veins and Origami Model of Digitate Leaves, Forma, 9, pp.253-257, 1994.
[8] K.Kaino, K.Yajima and N.Chiba, Origami Modeling Method of Leaves of Plants and CG Image Generation of Flower Arrangement, Proceedings of ICPADS 2000 Workshops, pp.207-212.

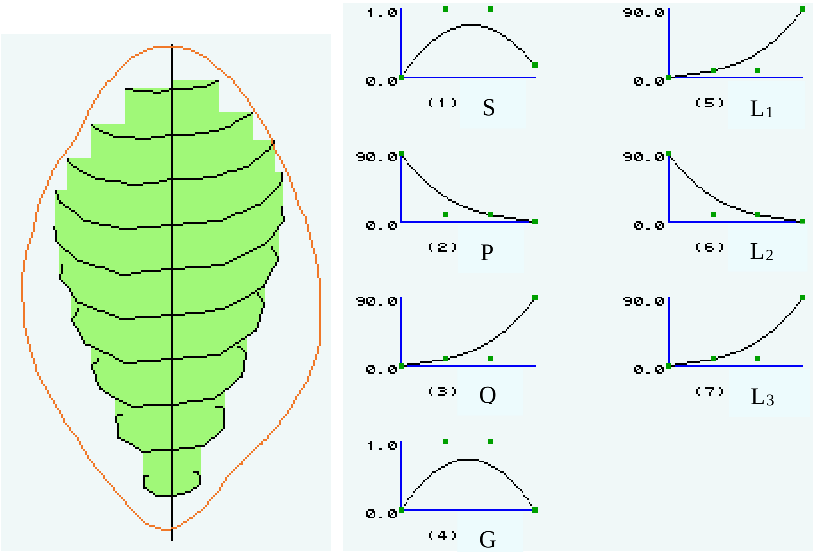


Figure 7: The best individual in the 1st generation.

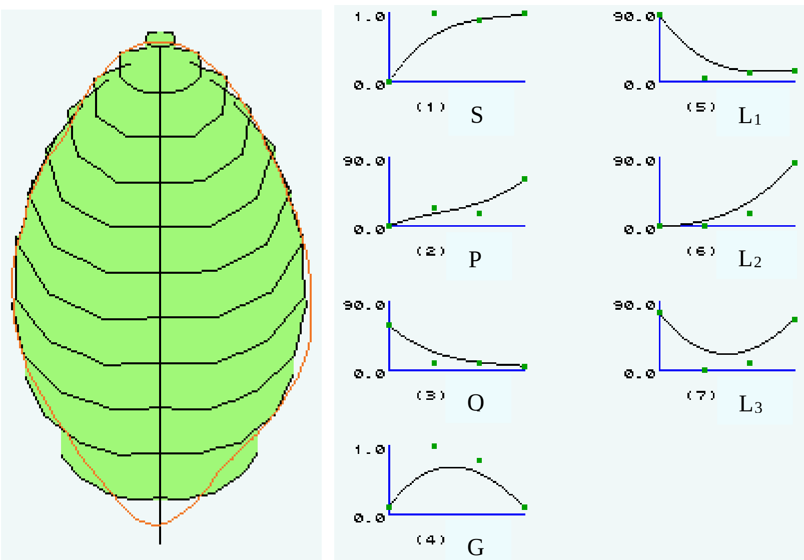
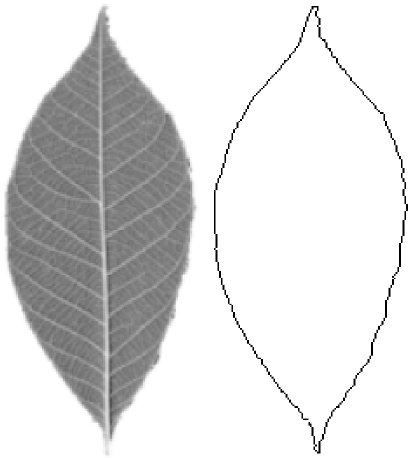


Figure 8: The best individual in the 200th generation.



(a) (b)

Figure 9: (a) A real leaf and (b) its outline.

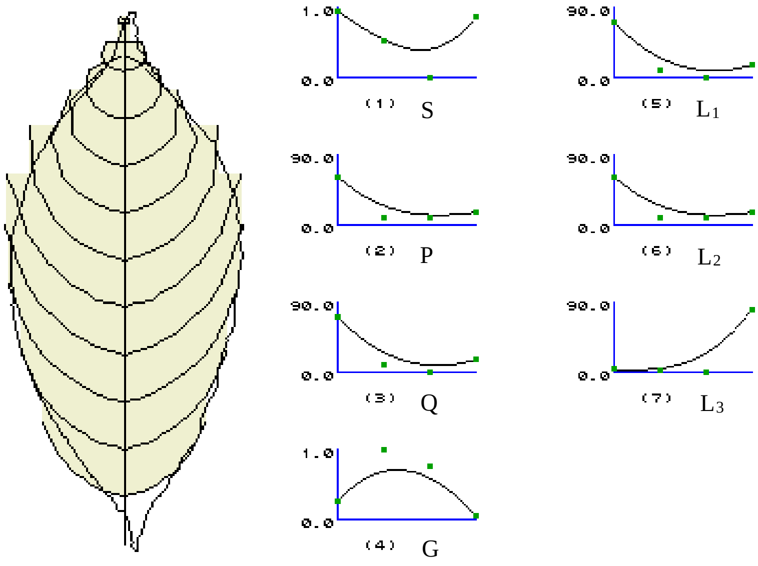


Figure 10: The best individual in the 200th generation.

Figure 11: CG images generated by using the obtained vein structure.