INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Mixed IFS : resolution of the inverse problem using Genetic Programming

Evelyne LUTTON , Jacques LEVY-VEHEL ,
Guillaume CRETIN , Philippe GLEVAREC , Cédric ROLL

N° 2631

August 1995

PROGRAMME 5

ISSN 0249-6399

INRIA
ROCQUENCOURT

Mixed IFS : resolution of the inverse problem using Genetic Programming

Evelyne LUTTON , Jacques LEVY-VEHEL ,
Guillaume CRETIN , Philippe GLEVAREC , Cédric ROLL

Programme 5 — Traitement du signal, automatique et productique
Projet FRACTALES

Rapport de recherche n ° 2631 — August 1995 — 17 pages

Abstract: We address here the resolution of the so-called inverse problem for IFS. This problem has already been widely considered, and some studies have been performed for affine IFS, using deterministic or stochastic methods (Simulated Annealing or Genetic Algorithm) [17, 10]. When dealing with non affine IFS, the usual techniques do not perform well, except if some *a priori* hypotheses on the structure of the IFS (number and type functions) are made. In this work, a Genetic Programming method is investigated to solve the "general" inverse problem, which permits to perform at the same time a numeric and a symbolic optimization. The use of "mixed IFS", as we call them, may enlarge the scope of some applications, as for example image compression, because they allow to code a wider range of shapes.

Key-words: Fractals, Genetic Programming, Inverse problem for IFS

(Résumé : tsvp)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

IFS mixtes : résolution du problème inverse par programmation génétique

Résumé : Nous nous intéressons ici à la résolution du problème inverse pour les IFS, largement étudié dans le cadre de la géométrie fractale. Celui-ci a déjà été assez bien résolu dans certains cas en ce qui concerne les IFS affines, par des méthodes déterministes ou stochastiques (recuit simulé ou algorithmes génétiques) [17, 10]. En revanche, si l'on souhaite aborder le problème général, c'est à dire mettant en jeu des IFS non affines, les techniques précédentes sont difficilement utilisables, sauf si l'on pose des hypothèses a priori sur la structure des IFS (nombre et type des fonctions). Nous proposons ici l'emploi d'une technique de programmation génétique pour la résolution du problème inverse général, qui permet d'effectuer simultanément une optimisation numérique et symbolique. La résolution du problème inverse pour les "IFS mixtes" pourra élargir le champ de certaines applications, comme par exemple la compression d'images, car ceux-ci permettent de coder une plus large variété de formes.

Mots-clé : Fractales, Programmation Génétique, Problème inverse pour les IFS

1 Introduction

IFS (Iterated Functions System) theory is an important topic in fractals. The geometric and measure theoretical aspects of systems of contractive maps (and associated probabilities) were worked out by J. Hutchinson [14], and the existence of a unique compact invariant set was proven. These studies have provided powerful tools for the investigation of fractal sets, and the action of systems of contractive maps to produce fractal sets has been considered by numerous authors (see for example [2, 3, 8, 12]).

A major challenge of both theoretical and practical interest is the resolution of the so called inverse problem [20, 26, 25, 4]. An exact solution can be found in some particular cases, but in general, no exact solution is known.

From a computational viewpoint this problem may be formulated as an optimization problem. A lot of work has been done in this framework, and some solutions exist, based on deterministic or stochastic optimization methods. Most of them make some *a priori* restrictive hypotheses : affine IFS, with a fixed number of functions [5, 15, 9, 27, 17]. Solutions based on Genetic Algorithms (GA) or Evolutionary Algorithms have recently been presented for affine IFS [25, 10, 24, 21].

As it will be seen in section 3, non-affine IFS provide an interesting variety of shapes, whose practical interest might be large. However, in that case, the inverse problem cannot be addressed using the "classical" techniques. We propose to make use of Genetic Programming in that framework. As far as we know, this is the first attempt to use Genetic Programming to solve that problem.

We will first recall IFS theory in section 2, then present some examples of mixed IFS attractors (section 3), and finally detail our genetic programming method (section 4).

2 IFS theory

An IFS (Iterated Function System) $\mho = \{F, (w_n)_{n=1,..,N}\}$ is a collection of N functions defined on a complete metric space (F, d).

Let W be the Hutchinson operator, defined on the space of subsets of F :

$$\forall\ K\ \subset\ F,\quad W(K) = \bigcup_{n\in[0,N]} w_n(K)$$

Then, if the w_n functions are contractive (the IFS is then called an *hyperbolic* IFS), there exists a unique set A such that :

$$W(A) = A$$

A is called the **attractor** of the IFS.

Recall : A mapping $w : F \to F$, from a metric space (F, d) into itself, is called **contractive** if there exists a positive real number $s < 1$ such that:

$$d(w(x), w(y)) \leq s.d(x, y) \quad \forall x, y \in F$$

The uniqueness of an hyperbolic attractor is a result of the Contractive Mapping Fixed Point Theorem for W, which is contractive according to the Hausdorff distance :

- *Hausdorff distance :*

$$d_H(A, B) = \max[\max_{x\in A}(\min_{y\in B} d(x, y)), \max_{y\in B}(\min_{x\in A} d(x, y))]$$

- *Contractive Mapping Fixed Point Theorem:*

  if (F, d) is a complete metric space, and $W : F \to F$ is a contractive transformation, then W has a unique fixed point.

From a computational viewpoint, an attractor can be generated according to two techniques:

- **Stochastic method (toss-coin)**
  Let x_0 be the fixed point of one of the w_i functions. We build the points sequence x_n as follows: $x_{n+1} = w_i(x_n)$, i being randomly chosen in $\{1..N\}$.
  Then $\bigcup_n x_n$ is an approximation of the real attractor of $\mho$. The larger n is, the more precise the approximation is.

- **Deterministic method :**
  From any kernel S_0, we build the sets sequence $\{S_n\}$

  $$S_{n+1} = W(S_n) = \bigcup_n w_n(S_n)$$

  When n tends to ∞ , S_n is an approximation of the real attractor of $\mho$.

The inverse problem for 2D IFS can be stated as follows:
for a given 2D shape (a binary image), find a set of contractive maps whose attractor resembles more this shape, in the sense of a pre-defined error measure.

Our error measure will be described in section 4.

$$w_1(x,y) = \begin{pmatrix} \sqrt{|\sin(\cos 0.90856 - \log(1+|x|))|} \\ \sin y \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \cos(\cos(\sqrt{|x|})) \\ \cos(\log(1+|y|)) \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \log(1+|\cos(\log(1+|y+x|))|) \\ \sqrt{|sin 0.084698|} \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} \log(1+|\sin(\sqrt{|0.565372|})|) \\ \sqrt{|0.81366 - ((\log(1+|0.814259|)) * \cos y)|} \end{pmatrix}$$
$$w_5(x,y) = \begin{pmatrix} \log(1+|\sqrt{|0.747399 + \cos y|}|) \\ \sin \frac{0.73624}{0.0001+|0.264553*y+0.581647+x|} \end{pmatrix}$$

Figure 1: A Mixed IFS, left, and its attractor, right.

3 Mixed IFS

In the case of affine IFS, each contractive map w_i of $\mho$ is represented as:

$$w_i(x,y) = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}$$

The inverse problem corresponds to the optimization of the values $(a_i, b_i, c_i, d_i, e_i, f_i)$ in order to get the attractor which resembles more the target.

When the w_i are not anymore restricted to be affine functions, we call the corresponding IFS **Mixed IFS**. The first point we have to address is the one of finding an adequate representation of these mixed IFS: the more natural one is to represent them as trees.

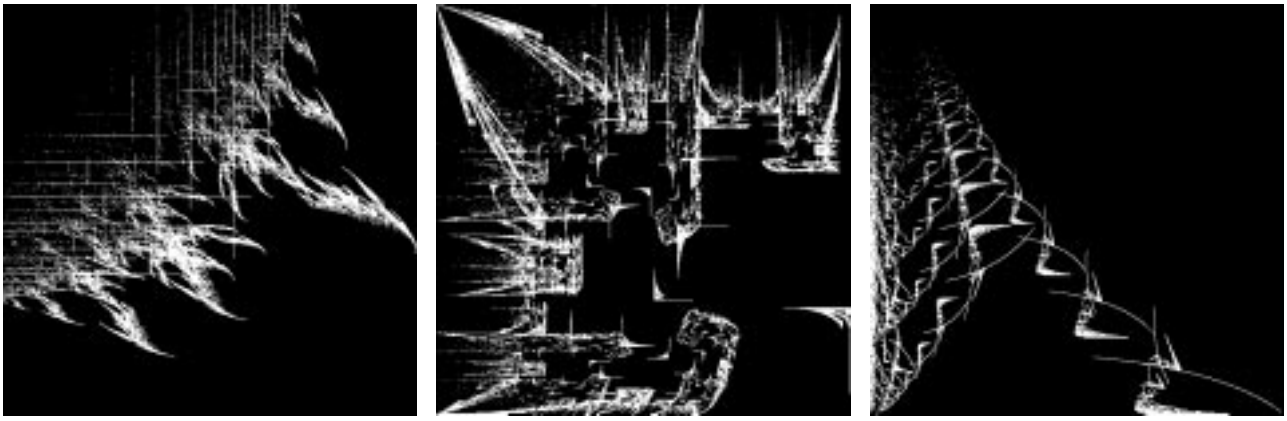

Figure 2: Other examples of attractors generated with mixed IFS.

The attractors of figures 1 and 2, are random mixed IFS: the w_i functions have been recursively built with help of random shots in a set of basic functions, a set of terminals (x and y), and a set of constants. In our examples, the constants belong to $[0, 1]$, and the basic functions set is:

- $+$
- $-$
- $\times$
- $div(x, y) = \frac{x}{0.0001+|y|}$
- cos
- sin
- $root(x) = \sqrt{|x|}$
- $loga(x) = \log(1 + |x|)$

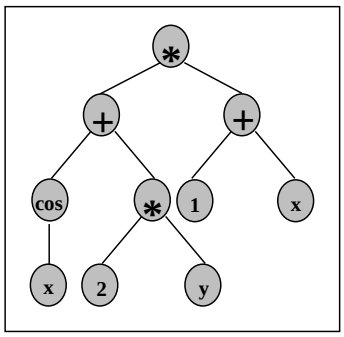

Figure 3: The function $((cos(x) + 2 * y) * (1 + x))$.

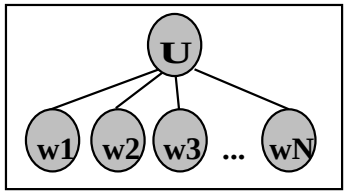

Figure 4: Representation of a mixed IFS.

We thus represent each w_i as a tree (see for example figure 3). The trees of the w_i are then gathered to build the main tree which represents the IFS $\mho$ (figure 4). This is a very simple structure which allows to code IFS with different numbers and different types of functions. The evaluation of such a structure is that of a simple mathematical expression evaluation. However, note that the evaluation is recursive, and thus may be time consuming.

As we have seen, generating a mixed IFS is done via simple recursive random shots. The set of possible IFS depends on the choice of the basic functions set and constants set. A difficult problem for mixed IFS is to verify that the w_i are contractive, in order to select *hyperbolic* IFS. On the contrary to affine IFS, this verification is not straightforward, and is in fact computationally intractable. We thus propose to use some heuristics that reject strongly non-contractive functions. The simplest way to do that (see section 4.3 for a finer criterion) is to verify the contractivity on some sample points, for example vertices of a grid placed on the domain.

Besides, as we have chosen to generate IFS whose attractors are in the $[0,1] \times [0,1]$ domain, we verify at the same time that each grid vertex remains in the domain.

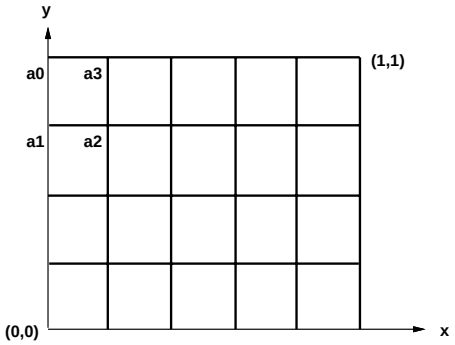

Figure 5: The domain constraint is tested on each vertex, the contractivity constraint, on each couple of vertices.

4 Genetic Programming to address the inverse problem

4.1 Introduction

Since John Koza [16] first proposed to extend the GA model to the space of computer programs, in order to create programs able to solve problems for which they haven't been explicitly programmed, a lot of very different applications have arisen : robotics, control, symbolic regression for example.

Compared to Genetic Algorithms approaches (GA), the individuals in a GP population are not any more strings of fixed length, but are programs that, when they are executed, give a possible solution to the problem. Typically, these programs are coded as trees.

The populations programs are built from elements of a set of functions and of a set of terminals which are typically symbols selected as being appropriate to the kind of problems we are solving. The "crossover" operation is performed by exchanging sub-trees between the programs and generally the "mutation" operation is not used in GP. When it is used, mutation consists in sometimes (with a weak probability) modifying a symbol of the tree.

The evolution of a program inside a GP algorithm is done simultaneously on its size, its structure and its content : the search space is the set of all recursively possible (sometimes according to some restriction rules) structures, built from the functions, terminal and constant sets (see figure 6),

When applying GP (or GA) to the resolution of a given problem, one generally has to deal with several points, namely :

- coding of the individuals,
- evaluation function of the individuals (fitness),

- *Generate an initial population of random compositions of the functions and the terminals of the problem (computer programs).*
- *Iteratively perform the following sub-steps until the termination criterion has been satisfied :*
  - *Execute each program in the population and assign to it a fitness value according to how well it solves the problem.*
  - *Create a new population of computer programs by applying the following two primary operations. The operations are applied to computer programs in the population chosen with a probability based on fitness (selection).*
    - *Copy some existing computer programs in the new population.*
    - *Create new computer programs by genetically recombining randomly chosen parts of two existing programs.*
- *The best computer program that appeared in any generation (i.e the best so far individual) is designated as the result of genetic programming. This result may be a solution (or an approximate solution) to the problem.*

Figure 6: Structure of a GP algorithm.

- definition of the genetic operators,
- choice of the parameters.

Concerning the first point, as we have already seen, the individuals of the population (i.e the Mixed IFS), are coded as trees. It allows to code a variable number of functions (dynamically), and it is an appropriate data structure for the mutation and the crossover.

In the following, we will address the other points, and insist on the original ones for our application : the use of two different types of mutation and the integration of the contractivity constraints in the fitness.

4.2 The fitness function

From a general viewpoint, the fitness function is a major procedure in GP or GA applications, because fitness is evaluated a large number of times at each generation. Moreover, in most complex problems, as the one we deal with, the fitness evaluation step is time consuming. For these reasons, the fitness evaluation procedure must be very carefully implemented : it can severely influence the computational time and results accuracy.

In our application, we have to characterize the quality of an IFS, that means to evaluate how far is its attractor from the target image.

4.2.1 Fitness based on Collage theorem versus fitness based on toss-coin algorithm

Among people dealing with inverse problem for IFS with GA, it is largely admitted that the fitness function based on the so-called collage theorem is preferable to a fitness based on a direct evaluation of the attractor

via the toss-coin algorithm. Indeed, the first method is very attractive and can be less time consuming than toss-coin evaluation algorithm.

Collage theorem : Let A be the attractor of the hyperbolic IFS $\mho = \{w_1, ..., w_n\}$:

$$\forall K \subset F, \qquad d_H(K, W(K)) < \varepsilon \qquad \Rightarrow \; d_H(K, A) < \frac{\varepsilon}{1-\lambda}$$

λ being the smallest number such that : $\forall n, \forall (x,y) \in F^2, \;\; d(w_n(x), w_n(y)) < \lambda . d(x,y)$

This theorem means that the problem of finding an IFS $\mho$, whose attractor is close to a given image I, is equivalent to the minimization of the distance :

$$d_H(I, \bigcup_{i=1}^{n} w_i(I))$$

under the constraint that the w_i are contractive functions.

But if $d_H(I, \bigcup_{i=1}^{n} w_i(I))$ is to be used as the fitness function in a GA (or a GP algorithm), then :

- The fitness depends on the contractivity of the maps ; if one of the maps is weakly contractive, then the term $\frac{1}{1-\lambda}$ may become very large, and the bound becomes meaningless. Moreover, in the case of affine IFS, it is possible to estimate λ and thus to minimize $\frac{1}{1-\lambda} d_H(I, \bigcup_{i=1}^{n} w_i(I))$ to overcome this difficulty. For mixed IFS, the contraction factor may not be uniform over the domain and is almost impossible to estimate.
- The Hausdorff distance itself is CPU-time consuming, and may also appear as counter-intuitive in many cases : on the figure 7 are represented two couples of shapes [(a), (b)] and [(a'), (b')] with d_H[(a), (b)] = d_H[(a'), (b')]. While (a) and (b) are perceived as similar, (a') and (b') look quite different.

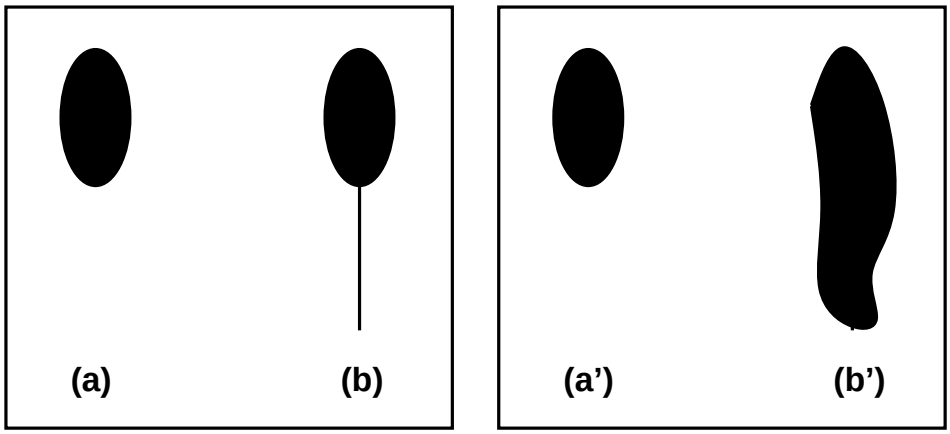

Figure 7: Hausdorff distance may be counter-intuitive.

These drawbacks led us to use the toss-coin fitness, which experimentally provides more precise results. Moreover, the direct computation of a distance between the target and the estimated attractor, computed using the toss-coin algorithms allows :

- to have variable accuracy estimations of the attractor, by tuning of the iterations number (see section 4.2.2 below),
- to use a more intuitive distance between shapes (namely pixels difference or quadratic distance), instead of the HausdorfF distance.

4.2.2 Practical fitness computation

In order to improve the algorithm efficiency, we have modified the fitness computation in two ways:

- As the fitness computation is the most computation time consuming procedure (it is repeated a large number of time), it must be considered very carefully. The toss-coin algorithm generally needs a lot of iterations to create the IFS's attractor. But as we noticed that the population quickly converges to a rough approximation of the target, only an approximation of the attractor may be needed at the beginning of the optimization process. We thus make the iteration number linearly increase during the generations, in order to provide a quickly computed approximation at the beginning of the GP, and then progressively tune fine details along the computation.
- In order to guide the research of the optimum, we use distance images. This allows to consider "smoother" functions to be optimized, as in [19]. A distance image is the transformation of a black & white image into a grey-level one, where the level affected to each image point is a function of its distance to the original shape. It can be easily computed by a simple algorithm (see [6]), based on the use of two masks (see figure 8): the resulting images are parameterized by $d1$ and $d2$ which represent the two elementary distances on vertical/horizontal and diagonal directions. This parameterization allows to use distances which are more or less "abrupt". For practical reasons, we use here grey level values which are proportional to the inverse of a distance. White pixels (value 255) are inside the attractor. Pixels get darker when their distance to the attractor increases (values between 254 and 0).

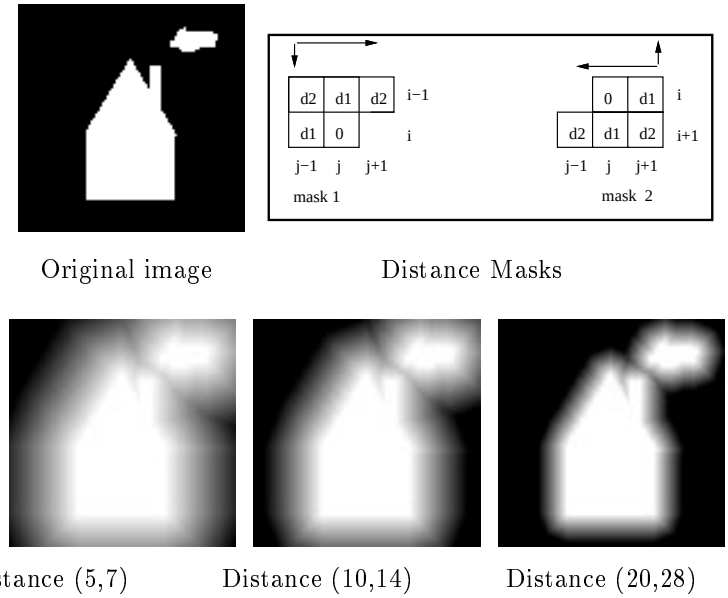

Figure 8:

The computation of the fitness of the current IFS is thus based on a measure of the difference between its attractor and the distance image of the target. The simple byte-to-byte difference (i.e. a counting of

coinciding white pixels) is thus completed by the mean value of the grey levels of the points belonging to the evaluated IFS attractor. This yields to the algorithm more "local" informations about the resemblance between the attractor and the target.

We improved this technique by varying the distance image parameters ($d1$ and $d2$) along the generations: we begin with a very fuzzy distance image. Every x generations we modify it so that at the end it becomes the real B&W attractor. Tolerance to small errors, and computation times have been thus improved.

4.3 Contractivity constraints

Before each individual evaluation, we have to verify if it is an hyperbolic IFS (thus yielding an unique attractor). As we have seen before, this verification is uneasy on mixed IFS, mainly because of the non linearity of the mappings. We have proposed in section 3 to simply verify the contractivity conditions on some sample points of the domain, and reject the individuals which does not verify it. This is a way to discard a lot of non-contractive IFS from the current population. But we have to notice that it may not discard some pathological mappings, even if we use a lot of sampling points.

We propose to address this problem in a different way, which will allow at the same time to use an *a priori* information in the target image, and to reduce the computation time. Our approach is based on the fixed point theorem.

For an hyperbolic IFS $\mho = \bigcup w_i$ whose attractor is A, each mapping w_i is contractive, and thus admits an unique fixed point X_i. We must then have :

$$\forall i, X_i \in A$$

The verification of the existence of the X_i's and their estimation can be easily performed : we built two suites of points $x^i_{n+1} = w_i(x^i_{n+1})$ starting from two points of the domain (for example (0,0) and (1,1)) :

1. Within a few iterations we can estimate the fixed point or decide that the function is not contractive. The use of two sequences allows to speed up the fixed points estimation.

2. We then check if the X_i's belongs to the target shape. This test yields a rough estimation of the chance of $\mho$ to correctly approximate I.

Notice that 1 only gives a necessary condition for the mapping to be contractive.

Practically, we compute a constraints function : $C(\mho)$ which is the mean distance value (measured on the distance image of the target) of the X_i's to the target. If $C(\mho)$ has too low a value, the fitness computation using the toss-coin algorithm can be pruned.

The fitness computation integrates the contractivity constraints in the following way :

1. *If there exists a w_i which is not contractive, then fitness$(\mho) = -1$ and the individual is directly discarded from the population.*
2. *If $C(\mho) < C_0$ then fitness$(\mho) = C(\mho)$*
3. *If $C(\mho) \geq C_0$ then the attractor A of $\mho$ is computed using the toss-coin algorithm, and fitness$(\mho)$ measures the difference between A and the target.*

4.4 Genetic operators

Crossover : we use the classical GP crossover which performs exchanges of randomly selected nodes between the parent-trees (see figure 9).

Mutation : we decided to use mutation in our algorithm, which is a common operator in GA, but a quite rare one in classical GP.

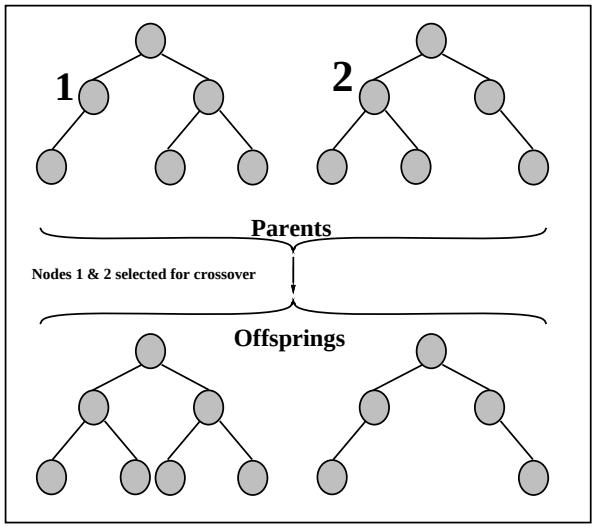

Figure 9: GP Crossover: nodes 1 and 2 are selected for crossover.

Indeed, mutation in a GA is a small change in the genetic code of the chromosome, for example, in the case of binary codes, mutation is a bit flip of one of the genes. In the case of GP, mutation has to slightly perturb a tree structure. In this view, we have to differentiate the nodes and the leaves of the tree:

- The nodes belong to the basic functions set, which is finite. A node mutation could be to replace one node by another basic function randomly chosen in the basic functions set. Since such a perturbation may have too drastic effects, we have preferred not to use it.
- The leaves are chosen in a terminals set (x or y) or in a constants set, which is a continuous interval ($[0,1]$). We also have to separate the mutation of constants to the mutation of variables, because they are of different nature. Of course we could also imagine a mutation process which transforms a constant into a variable and reversely. However, it seems to be too violent, except in the case of variables, as we will see.
  - *Constants*: mutation is the only mean to make constants evolve. This is very important in our case, because we need to perform a numerical optimization of the constants. We perturb the constants with a parameterized probability (see the parameters summary, section 4.5). A constant is replaced by a new value obtained from an uniform random shot inside a disk of fixed radius (another parameter of our algorithm) around it (see figure 10).
  - *Variables*: an "internal" mutation, i.e. changing a x to a y and reversely is again possible, but we preferred a mutation which changes a variable into a randomly chosen constant (see figure 11). We have made this choice on an empirical basis: we noticed that in some cases constants tend to disappear from the current population. Once they have disappeared, they cannot reappear in the offsprings populations. We thus propose to use a constants creation process, via mutation of variables, to maintain a minimal proportion of constants in the population.

    The constants vanishing effect we have experimentally noticed may be explained as follows: the numerical optimization of the constants is a more difficult task than the symbolic optimization of the other nodes. The selection operator thus tends to eliminate too rapidly IFS having bad constants. This difference is due to the fact that the search spaces of the nodes and variables is a finite one while the search space of the constants is theoretically infinite.

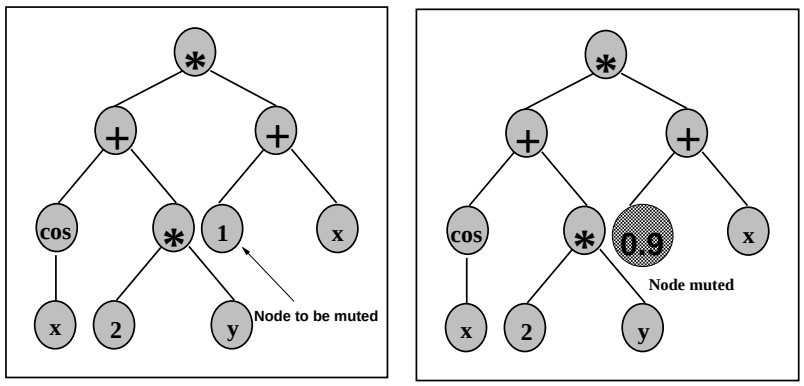

Figure 10: Mutation of constants.

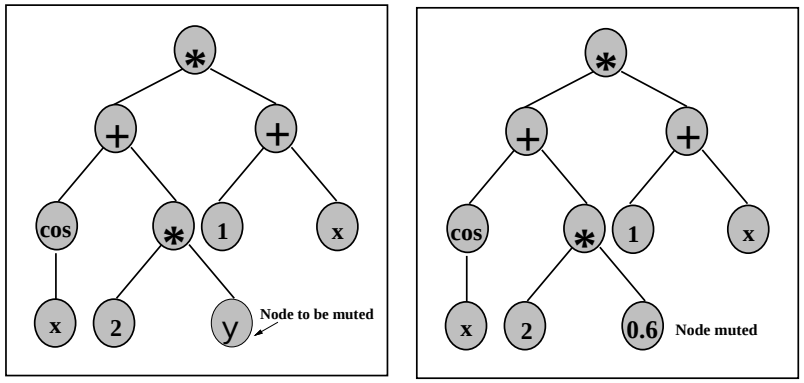

Figure 11: Mutation of variables.

Other techniques (that we have not tested) that allow to avoid the constants to disappear, may be to reduce the size of the constants search space by allowing only a finite set of constants (via sampling for example), or separate the symbolical and the numerical optimization (i.e. having a subprocess which optimizes the constants before each IFS evaluation).

4.5 Parameters setting

As it has be seen before, there are a lot of parameters that have to be tuned to make the algorithm efficient. We here summarize these parameters, and precise the practical settings for each :

Image size : the method was tested on images from 64x64 to 256x256 pixels.

Population size : Typically 20 to 50 individuals, bigger populations were less efficient.

Maximum number of generations : Typically 1000 to 2000 : as small populations sizes are used, a large number of generations is needed in order to converge. This approach is more efficient than an algorithm with a large population size and a smaller number of generations.

Crossover probability : Typically 0.7 to 0.9.

Mutations probabilities : Typically 0.1 to 0.2 for the constants, and 0 to 0.01 for the variables.

Range of the constants : [0,1].

Perturbation radius of the constants during a mutation : between 0.05 and 0.15. The mutation of a constant is thus a uniform random shot inside an interval centered on the constant.

Maximum and minimum allowed number of contractive maps in the mixed IFS : from 3 to 7 maps. This is the only constraints set on the structures of the evolved IFS's trees. No depth restrictions are imposed. However, we experimentally verified that their structures do not excessively expand along the evolution.

5 Results

We have tested our algorithm on shapes that were actual attractors of IFS, some generated with randomly chosen contractive maps. The choice of basic functions for the GP is the one presented in section 3. Initial populations are randomly chosen.

We present here three good convergence results. For each example, we present : the target attractor, the best image obtained after convergence, the fitness evolution curve, the parameters setting, and the functions composing the best IFS, compared to the "true" ones (in general, there is an infinite number of IFS leading to the same attractor).

- **Example #1** : approximation of a square, see figures 12, 13 and table 1 for the parameters setting.

  IFS of the best image :

$$w_1(x,y) = \begin{pmatrix} \sin x \\ \sin(\sin(\cos(\sin y))) \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \sin(\sin x) \\ \sin y \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \sin x \\ \sin(\sin y) \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} \sin(\sin(\cos x)) \\ \sin y \end{pmatrix}$$
$$w_5(x,y) = \begin{pmatrix} \sin(\sin x) \\ \sin(\sin y) \end{pmatrix}$$

  IFS of the target image :

$$w_1(x,y) = \begin{pmatrix} 0.5x + 0.5 \\ 0.5y + 0.5 \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} 0.5x - 0.5 \\ 0.5y + 0.5 \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} 0.5x + 0.5 \\ 0.5y - 0.5 \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} 0.5x - 0.5 \\ 0.5y - 0.5 \end{pmatrix}$$

- **Example #2** :approximation of a random IFS, see figures 14, 15 and table 2 for the parameters setting.

  IFS of the best image :

$$w_1(x,y) = \begin{pmatrix} \cos x \\ \cos(\cos(\cos y)) \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \sin x * \cos(\sin y) \\ \cos(\sin y) \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \sin x \\ \cos y \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} \sin(\sin x) \\ \log(1 + |y|) \end{pmatrix}$$
$$w_5(x,y) = \begin{pmatrix} \sin(\sin(\sin x)) * \cos(\cos x) \\ \sin(\sin y) \end{pmatrix}$$

  IFS of the target image :

$$w_1(x,y) = \begin{pmatrix} \sin x \\ \cos y \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \log(1 + |x|) \\ \cos y \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \cos x \\ \sin y \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} \sqrt{|\sin(\log(1 + \log(1 + |x|))) - \sin(\sin x - 0.118226)|} \\ \cos(\sqrt{|(y * x - \sin x) * \sin y|}\,) \end{pmatrix}$$

- **Example #3**: approximation of a random IFS, see figures 16, 17 and table 3 for the parameters setting.
  IFS of the best image:

$$w_1(x,y) = \begin{pmatrix} \sin x \\ \cos y \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \log(1+|x|) \\ \cos y \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \cos x \\ \sin y \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} \sqrt{(|\sin(\log(1+\log(1+|x|))) - (\sin((\sin x) - 0.118226))|)} \\ \cos(\sqrt{(|((y*x) - (\sin x)) * (\sin y)|)}) \end{pmatrix}$$

  IFS of the target image:

$$w_1(x,y) = \begin{pmatrix} \sin x \\ \cos y \end{pmatrix}$$
$$w_2(x,y) = \begin{pmatrix} \log(1+\sqrt{|\cos(\sin x) - \sin(\cos(0.568514 - \cos(\sqrt{|\cos y|}))))|}) \\ \sqrt{|\cos(\cos(\sin y) - y * \cos(\ cos(\sqrt{\sqrt{y - \cos(\cos(\cos(\cos(\sin(\cos 0.473744)))))))}})|} \end{pmatrix}$$
$$w_3(x,y) = \begin{pmatrix} \cos x \\ \sin y \end{pmatrix}$$
$$w_4(x,y) = \begin{pmatrix} x * \sqrt{\sqrt{|0.335979 - \cos\sqrt{|x|}|}} \\ \cos y \end{pmatrix}$$
$$w_5(x,y) = \begin{pmatrix} \sin(\sqrt{|\cos y|} + x - \cos(\cos(\sin x))) \\ \cos(\cos(\cos(\cos y))) \end{pmatrix}$$

The first point to remark is that the functions of the approximations does not resemble the one of the target images (especially for the example #1): this is due to the fact that the representation of an attractor by a set of functions is not unique.

The parameters adjustment remains an uneasy task, but we empirically noticed the following facts:

- The distance images are very efficient. It is particularly obvious on the fitness evolution curves (figures 13, 15 and even more on figure 17): when updating the distance image, the curve suddenly falls down and then grows up again. For the new distance image, the value of the fitness becomes lower, because it is computed on a distance image with has larger d_1 and d_2 parameters. This corresponds in fact to a more precise evaluation of the difference between the current IFS and the target.
- The mutation of the constants is important, it brings diversity and cannot be set to zero.

Finally, the target images which yield good results are rather compact: the convergence to line-shaped targets is more difficult.

Figure 12: Example #1, from left to right : original image and best images of generations 10, 100, 300 and 1500.

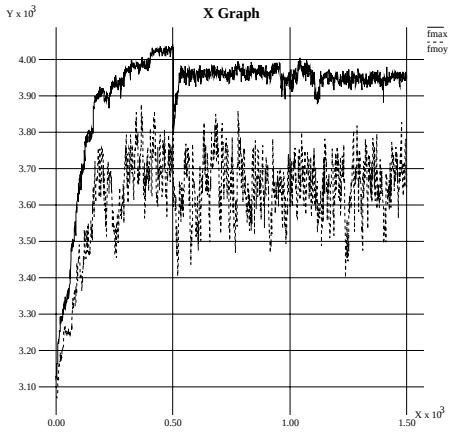

Figure 13: Example #1 : fitness evolution. The maximum fitness of the current population is the continuous curve, the mean fitness is the dotted one.

Image size	64 pixels
Population size	30
Max number of generations	1500
Crossover probability	0.7
Mutation probability for constants	0.2
Mutation probability for variables	0
Range of the constants	[0,1]
Perturbation radius for the constants	0.1
Max and min number of contractive maps	3 to 6

Table 1: Example #1 : parameters setting.

Figure 14: Example #2, from left to right : original image and best images of generations 50, 300, 500 and 1000.

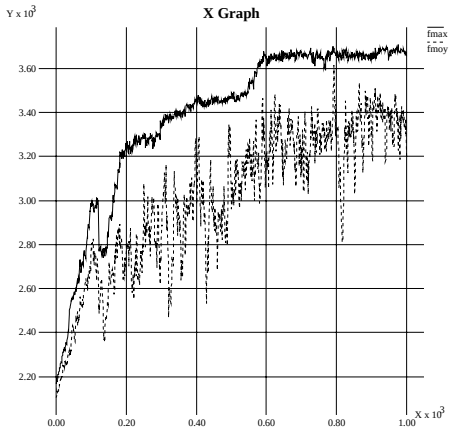

Figure 15: Example #2: fitness evolution. The maximum fitness of the current population is the continuous curve, the mean fitness is the dotted one.

Image size	64 pixels
Population size	20
Max number of generations	1000
Crossover probability	0.7
Mutation probability for constants	0.2
Mutation probability for variables	0
Range of the constants	[0,1]
Perturbation radius for the constants	0.1
Max and min number of contractive maps	4 to 6

Table 2: Example #2: parameters setting.

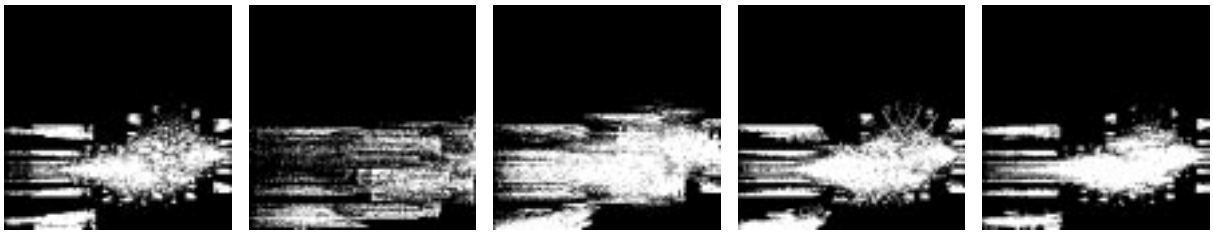

Figure 16: Example #3, from left to right : original image and best images of generations 50, 260, 1010 and 1300.

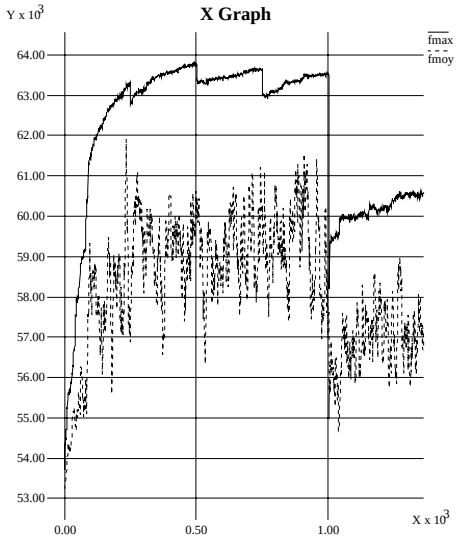

Figure 17: Example #3 : fitness evolution. The maximum fitness of the current population is the continuous curve, the mean fitness is the dotted one.

Image size	256 pixels
Population size	30
Max number of generations	1300
Crossover probability	0.85
Mutation probability for constants	0.25
Mutation probability for variables	0.001
Range of the constants	[0,1]
Perturbation radius for the constants	0.1
Max and min number of contractive maps	4 to 7

Table 3: Example #3 : parameters setting.

6 Conclusion

We have proposed a method to solve the "general" inverse problem for mixed IFS within a reasonable computation time (a few hours on Sparc 10 and Dec 5000 stations). This computation time is similar to computation times of GA applied to the inverse problem for affine IFS [18], although in the case of mixed IFS the size of the search space is much more larger. This fact may be explained by the use of variable sized structures in the GP algorithm, which seems to perform a more efficient search in a large space.

The method may be improved in several directions:

- test a "smoother" transition between distance images: a re-computation of distances images at every generations would allow to let the parameters d_1 and d_2 vary more smoothly,
- test other mutation strategies, as suggested in section 4.4,
- test an adaptive radius for mutation of constants, in the same way as for evolutionary programming techniques, where mutation variance is dynamically adapted, in function of the performance of the individual,
- make the iteration number of the toss coin evaluation algorithm be more adaptive (we can theoretically fix the iterations number and the probabilities of the toss coin algorithm in order to more rapidly approximate the attractor within a fixed error),
- modify the storage structure of the IFS in order to reduce the computation time (mainly by avoiding some useless computations)

Such an approach might be interesting in the field of image compression. IFS compression techniques are generally based on affine IFS. The use of mixed IFS may yield more flexible spatial and grey-level transformations, and thus may allow to improve the compression ratio for the same number of functions.

References

[1] J. Albert, F. Ferri, J. Domingo, and M. Vincens. An Approach to Natural Scene Segmentation by Means of Genetic Algoritms with Fuzzy Data . In *Pattern Recognition and Image Analysis*, pages 97–113, 1992. Selected papers of the 4th Spanish Symposium (Sept 90), Perez de la Blanca Ed.

[2] M. Barnsley and S. Demko. Iterated function system and the global construction of fractals. *Proceedings of the Royal Society*, A 399:243–245, 1985.

[3] M. Barnsley, S. Demko, J. Elton, and Geronimo J. Invariant measures for Markov processes arising from iterated function systems with place-dependent probabilities. *Georgia Tech. preprint.*

[4] M. Barnsley, V. Ervin, D. Hardin, and J. Lancaster. Solution of an inverse problem for fractals and other sets. *Proc. Natl. Acad. Sci. USA*, 83, 1986.

[5] M. F. Barnsley. Fractals Everywhere. *Academic Press,N Y*, 1988.

[6] G. Borgefors. Distance transformation in arbitrary dimension. *Computer Vision, Graphics, and Image Processing*, (27), 1984.

[7] Y. Davidor. Genetic Algorithms and Robotics. A Heuristic Strategy for Optimization. *World Scientific*, 1991.

[8] J. H. Elton. An ergodic theorem for iterated maps. *Georgia Tech. preprint*, 1986.

[9] Y. Fisher. Fractal image compression. *Siggraph 92 course notes*, 1992.

[10] B. Goertzel. Fractal image compression with the genetic algorithm. *Complexity International*, (1), 1994.

[11] D. A. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison-Wesley*, January 1989.

[12] D. P. Hardin. Hyperbolic Iterated Function Systems and Applications. *PhD thesis, Georgia Institute of Technology*, 1985.

[13] A. Hill and C. J. Taylor. Model-Based Image Interpretation using Genetic Algorithms . *Image and Vision Computing*, 10(5):295–301, June 1992.

[14] J. Hutchinson. Fractals and self-similarity. *Indiana University Journal of Mathematics*, 30:713–747, 1981.

[15] A.E. Jacquin. Fractal image coding: a review. *Proceedings of the IEEE*, 81(10), october 1993.

[16] J. R. Koza. *Genetic Programming* . MIT Press, 1992.

[17] J. Lévy Véhel. *Analyse et synthèse d'objets bi-dimensionnels par des méthodes stochastiques.* PhD thesis, Université de Paris Sud, Decembre 1988.

[18] E. Lutton and J. Levy-Vehel. Optimization of fractal functions using genetic algorihms. *research report INRIA*, (1941), June 1993.

[19] E. Lutton and P. Martinez. A genetic algorithm for the detection of 2d geometric primitives in images. In *12-ICPR*, 1994. Jerusalem, Israel, 9-13 October.

[20] G. Mantica and A. Sloan. Chaotic optimization and the construction of fractals : solution of an inverse problem, *Complex Systems* . (3):37–62, 1989.

[21] D. J. Nettleton and R. Garigliano. Evolutionary algorithms and a fractal inverse problem . *Biosystems*, (33):221–231, 1994. Technical note.

[22] G. Roth and M. D. Levine. Geometric Primitive Extraction Using a Genetic Algorithm . In *IEEE Computer Society Conference on CV and PR*, pages 640–644, 1992.

[23] S. Truvé. Using a Genetic Algorithm to solve Constraint Satisfation Problems Generated by an Image Interpreter. In *Theory and Applications of Image Analysis : 7th Scandinavian Conference on Image Analysis*, pages 378–386, August 1991. Aalborg (DK).

[24] L. Vences and I. Rudomin. Fractal compression of single images and image sequences using genetic algorithms. *The Eurographics Association*, 1994.

[25] E. R. Vrscay. Fractal Geometry and Analysis, *Chapter Iterated function Systems: theory, applications and the inverse problem*, pages 405–468. 1991.

[26] R. Vrscay. Moment and collage methods for the inverse problem of fractal construction with iterated function systems . In *Fractal 90 Conference*, 1990. Lisbonne, June 6-8.

[27] E.W. Jacobs Y. Fisher and R.D. Boss. Fractal image compression using iterated transforms. *Data Compression*, 1992.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399