*Complex Systems* **9** (1995) 375–398

# Mixed IFS: Resolution of the Inverse Problem using Genetic Programming

**Evelyne Lutton**
**Jacques Levy-Vehel**
**Guillaume Cretin**
**Philippe Glevarec**
**Cidric Roll**
*INRIA - Rocquencourt**
*B.P. 105, 78153 LE CHESNAY Cedex, France*

**Abstract.** We address here the resolution of the so-called inverse problem for the iterated functions system (IFS). This problem has already been widely considered, and some studies have been performed for the affine IFS, using deterministic or stochastic methods (simulated annealing or genetic algorithm). In dealing with the nonaffine IFS, the usual techniques do not perform well unless some *a priori* hypotheses on the structure of the IFS (number and type of functions) are made. In this work, a genetic programming method is investigated to solve the "general" inverse problem, which allows the simultaneous performance of a numeric and a symbolic optimization. The use of a "mixed IFS" may enlarge the scope of some applications, for example, image compression, because it allows a wider range of shapes to be coded.

## 1. Introduction

Iterated functions system (IFS) theory is an important topic in fractals. The geometric and measure theoretical aspects of systems of contractive maps (and associated probabilities) were worked out in [14], and the existence of a unique compact invariant set was proved. Such 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 (e.g., [2, 3, 8, 12]). A major challenge of both theoretical and practical interest is the resolution of the so-called inverse problem [4, 20, 25, 26]. Except for some particular cases, no exact solution is known. From a computational viewpoint this problem may be formulated as an optimization

*Electronic address: `http://www-rocq.inria.fr/fractales/`.

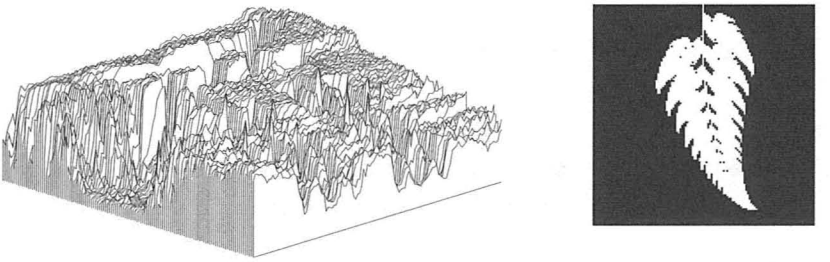

Figure 1: A two-dimensional slice of the error function for the Barnsley fern (left), for affine IFS (right). The dimension of the search space is 24.

problem. A lot of work has been done in this framework, and some solutions exist based on deterministic or stochastic optimization methods. As the function to be optimized is extremely complex (see Figure 1), most of them make some *a priori* restrictive hypotheses: use of an affine IFS, with a fixed number of functions [5, 9, 15, 17, 27]. Solutions based on genetic algorithms (GAs) or evolutionary algorithms, have recently been presented for the affine IFS [10, 21, 24, 25]. As seen in section 3, the nonaffine IFS provides an interesting variety of shapes, whose practical interest might be large. However, in this 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 this problem.

We first review 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. Iterated functions system theory

An IFS $\mho = \{F, (w_n)_{n=1,..,N}\}$ 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 $F$:

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

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

$$W(A) = A.$$

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

**Note.** A mapping $w : F \rightarrow F$, 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 a hyperbolic attractor is a result of the contractive mapping fixed-point theorem for $W$, which is contractive according to the Hausdorff distance as follows.

- *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 $W : F \rightarrow F$ 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 point 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 set sequence $\{S_n\}$,

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

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

The inverse problem for a two-dimensional IFS can be stated as follows.

> For a given two-dimensional shape (a binary image), find a set of contractive maps whose attractor more resembles this shape, in the sense of a predefined error measure.

Our error measure will be described in section 4.

$$w_1(x,y) = \begin{pmatrix} \sqrt{|\sin(\cos 0.90856 - \log(1+|x|))|} \\ \sin y \end{pmatrix}$$

$$w_2(x,y) = \begin{pmatrix} \cos(\cos(\sqrt{|x|})) \\ \cos(\log(1+|y|)) \end{pmatrix}$$

$$w_3(x,y) = \begin{pmatrix} \log(1+|\cos(\log(1+|y+x|))|) \\ \sqrt{|\sin 0.084698|} \end{pmatrix}$$

$$w_4(x,y) = \begin{pmatrix} \log(1+|\sin(\sqrt{|0.565372|})|) \\ \sqrt{|0.81366 - ((\log(1+|0.814259|)) * \cos y)|} \end{pmatrix}$$

$$w_5(x,y) = \begin{pmatrix} \log(1+|\sqrt{|0.747399 + \cos y|}|) \\ \sin\left(\frac{0.73624}{0.0001+|0.264553*y+0.581647+x|}\right) \end{pmatrix}$$

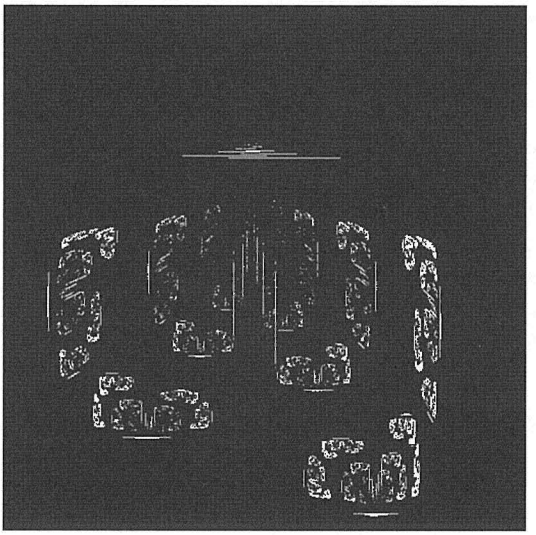

Figure 2: A mixed IFS and its attractor.

## 3. Mixed iterated functions system

In the case of an 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 ($a_i$, $b_i$, $c_i$, $d_i$, $e_i$, $f_i$) to get the attractor that more resembles the target. When the $w_i$ are no longer restricted to be affine functions, we call the corresponding IFS a *mixed IFS*. The first point we have to address is finding an adequate

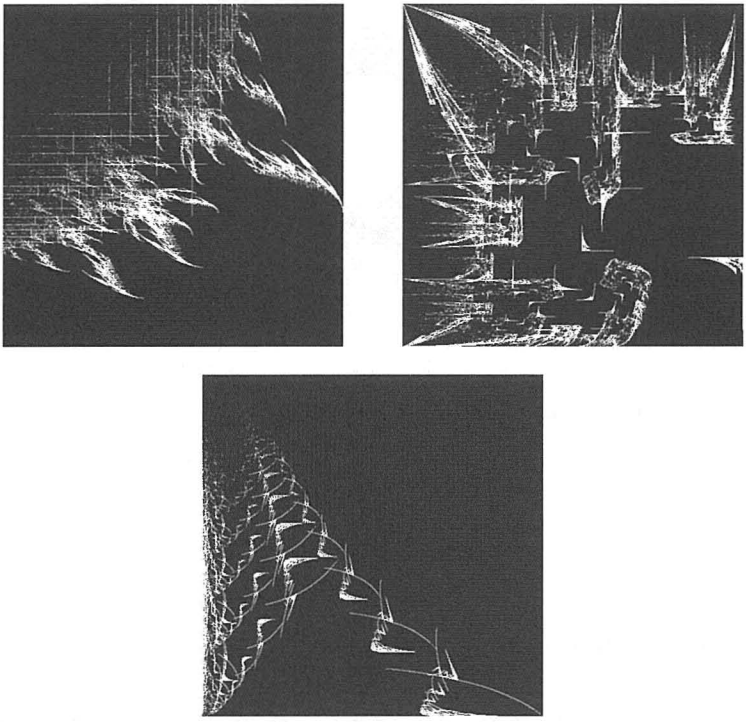

Figure 3: Other examples of attractors generated with mixed IFS.

representation of these mixed IFS. The more natural method is to represent them as trees. The attractors of Figures 2 and 3 are random mixed IFS. The $w_i$ functions have been recursively built with the 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 set of basic functions is

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

We thus represent each $w_i$ as a tree as shown in Figure 4. The trees of the $w_i$ are then gathered to build the main tree representing the IFS $\mho$ as shown in Figure 5. This is a very simple structure that allows an IFS to be coded with different numbers and different types of functions. The evaluation of such a structure corresponds to that of a simple mathematical expression. However, note that the evaluation is recursive and thus may be time consuming. As we

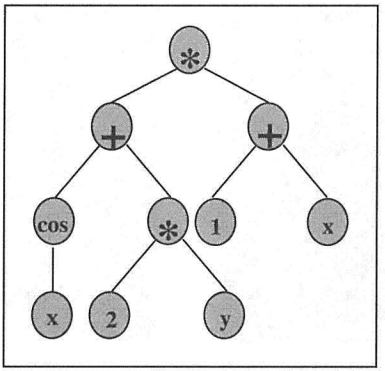


Figure 4: The function $((\cos(x) + 2 * y) * (1 + x))$.

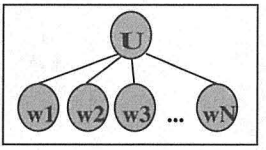


Figure 5: Representation of a mixed IFS.

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 function set and a constant set. A difficult problem for a mixed IFS is to verify that the $w_i$ are contractive, in order to select a *hyperbolic* IFS. Contrary to the case for an affine IFS, this verification is not straightforward for a mixed IFS and is, in fact, computationally intractable. We thus propose to use some heuristics that reject strongly noncontractive functions. The simplest way to do this (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. Because we have chosen to generate IFSs 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.

## 4. Genetic programming to address the inverse problem

### 4.1 Introduction

Since the first proposal to extend the GA model to the realm of computer programs [16], to create programs able to solve problems for which they were not explicitly designed, a lot of very different applications have arisen; for example, robotics control and symbolic regression. Compared with the GA approaches, the individuals in a genetic programming (GP) population are not strings of fixed length but are programs that, when executed, give a

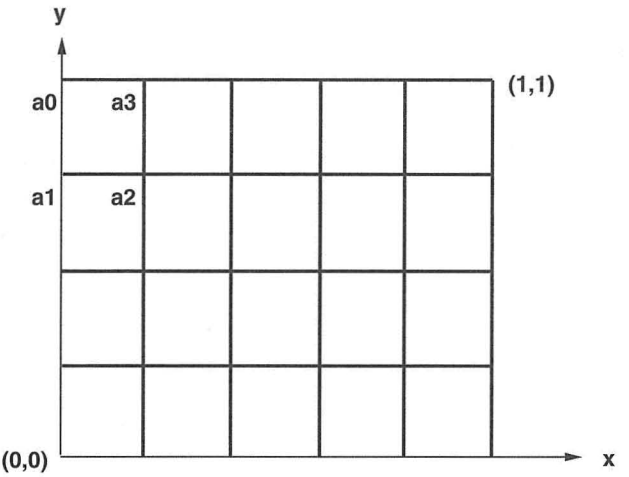


Figure 6: The domain constraint is tested on each vertex, and the contractivity constraint on each couple of vertices.

possible solution to the problem. Typically, these programs are coded as trees. The population programs are built from elements of a set of functions and a set of terminals that are typically symbols selected as being appropriate to the kind of problems being solved. The "crossover" operation is performed by exchanging subtrees between the programs. Generally, the "mutation" operation is not used in GP. When it is used, mutation sometimes (with a weak probability) involves modifying a symbol of the tree. The evolution of a program within 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 function, terminal, and constant sets (see Figure 7).

When applying GP (or GA) to the resolution of a given problem, one generally has to deal with several points such as the following.

- Coding of the individuals.
- Evaluation function of the individuals (fitness).
- 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. This allows the coding of a variable number of functions (dynamically), and it is an appropriate data structure for mutation and crossover. In the next section we 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 function.

1. *Generate an initial population of random compositions of the functions and the terminals of the problem (computer programs).*
2. *Iteratively perform the following substeps until the termination criterion has been satisfied.*
   a. *Execute each program in the population and assign to it a fitness value according to how well it solves the problem.*
   b. *Create a new population of computer programs by applying the following two primary operations to selected computer programs. This selection is done by choosing programs in the population with a probability proportional to their fitness.*
      i. *Copy some existing computer programs in the new population (with probability* $1 - p_c$*).*
      ii. *Create new computer programs by genetically recombining randomly chosen parts of two existing programs (with probability* $p_c$*).*
3. *The best computer program that appears 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 7: Structure of a genetic programming algorithm.

### 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, such 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, as it can severely influence the computational time and result accuracy. In our application, we have to characterize the quality of an IFS, that is, to evaluate how far its attractor is from the target image.

#### 4.2.1 Fitness based on collage theorem versus fitness based on toss-coin algorithm

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

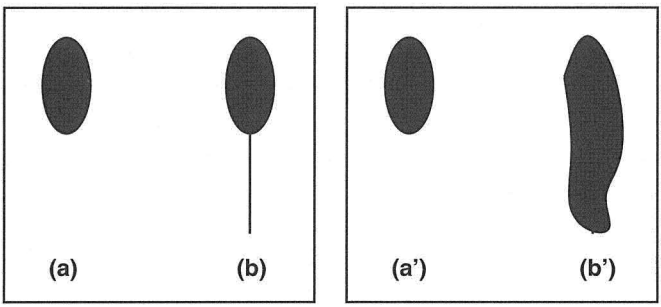


Figure 8: Hausdorff distance may be counterintuitive.

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

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

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

where $\lambda$ is 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 we have the following.

- The fitness depends on the contractivity of the maps; if one of the maps is weakly contractive, then the term $1/(1-\lambda)$ may become very large, and the bound becomes meaningless. Moreover, in the case of an affine IFS, it is possible to estimate $\lambda$ and thus to minimize $1/(1-\lambda) d_H(I, \bigcup_{i=1}^{n} w_i(I))$ 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 counterintuitive in many cases. For example, Figure 8 shows 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.

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 using the toss-coin algorithm allows the following.

- Variable accuracy estimations of the attractor, by tuning of the iterations number (see section 4.2.2).
- Use of a more intuitive distance between shapes (namely, pixel difference or quadratic distance) instead of the Hausdorf distance.

### 4.2.2 Practical fitness computation

To improve the algorithm efficiency, we have modified the fitness computation in the following two ways.

- As the fitness computation is the most computationally time-consuming procedure (it is repeated a large number of times), it must be considered very carefully. The toss-coin algorithm generally needs a lot of iterations to create the IFS's attractor. But because 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. This is done in order to provide a quickly computed approximation at the beginning of the GP, and then progressively fine-tune details during the computation.
- To guide the research of the optimum, we use distance images. This allows the consideration of "smoother" functions to be optimized, as in [19]. A distance image is the transformation of a black and white image into a gray-level image. The level assigned to each image point is a function of its distance from the original shape. It can be easily computed using a simple algorithm (see [6]), based on the use of two masks shown in Figure 9. The resulting images are parameterized by $d_1$ and $d_2$, which represent the two elementary distances in vertical/horizontal and diagonal directions. This parameterization allows the use of distances that are more or less abrupt. For practical reasons, here we use gray-level values that are proportional to the inverse of a distance. White pixels (value 255) are inside the attractor. Pixels get darker as their distance to the attractor increases (values between 254 and 0).

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

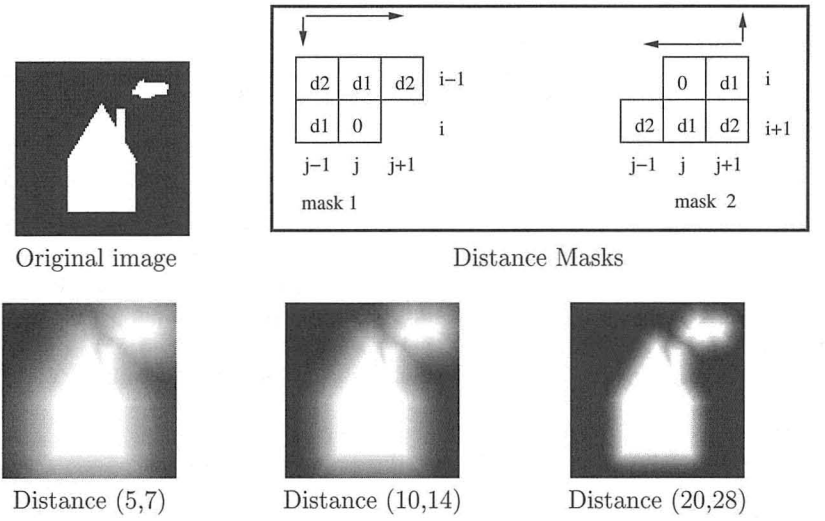


Figure 9: Transforming an image from black and white to gray-levels.

We improved this technique by varying the distance image parameters ($d_1$ and $d_2$) 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 black and white attractor. Tolerance to small errors and computation times have thus been improved.

### 4.3 Contractivity constraints

Before each individual evaluation, we have to verify that it is a hyperbolic IFS (thus yielding a unique attractor). As we have seen, this verification is not easy for a mixed IFS, mainly because of the nonlinearity of the mappings. We have proposed simply verifying the contractivity conditions on some sample points of the domain, and rejecting the individuals for which the conditions are not verified. This is a way to discard a lot of noncontractive IFS from the current population. But it may not discard some pathological mappings, even if we use a lot of sampling points. We propose addressing this problem in a different way, which will allow us to use *a priori* information in the target image and to reduce the computation time. Our approach is based on the fixed-point theorem. For a hyperbolic IFS $\mho = \bigcup w_i$ whose attractor is $A$, each mapping $w_i$ is contractive and thus admits a unique fixed point $X_i$. We must then have

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

The verification of the existence of the $X_i$ 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)).

- Within a few iterations we can estimate the fixed point or decide that the function is not contractive. The use of two sequences allows us to speed up the fixed-point estimation.
- We then check if the $X_i$ point belongs to the target shape. This test yields a rough estimation of the chance that $\mho$ correctly approximates $I$.

Notice that the first step above only gives a necessary condition for the mapping to be contractive. Practically, we compute a constraint function $C(\mho)$ which is the mean distance value (measured on the distance image of the target) of the $X_i$ 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 ways.

- If there exists a $w_i$ that is not contractive, then fitness$(\mho) = -1$ and the individual is directly discarded from the population.
- If $C(\mho) < C_0$, then fitness$(\mho) = C(\mho)$.
- 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 that performs exchanges of randomly selected nodes between the parent trees (see Figure 10).

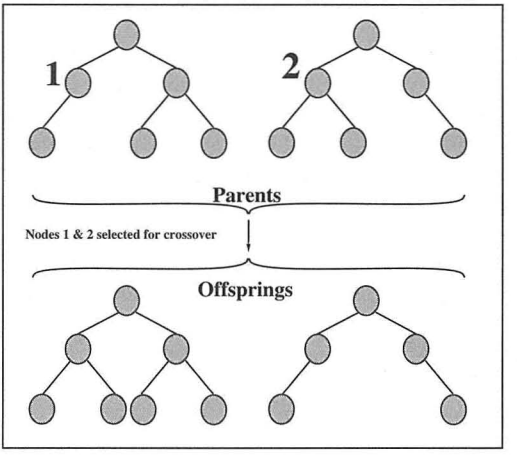


Figure 10: GP crossover with nodes 1 and 2 selected for crossover.

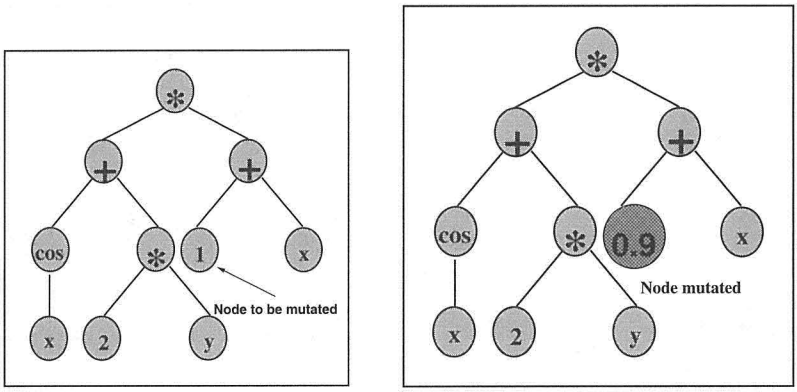


Figure 11: Mutation of constants.

*Mutation.* We decided to use mutation in our algorithm, which is a common operator in GA but seldom used in classical GP. 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.

1. *Nodes.* The nodes belong to the basic function set, which is finite. A node mutation could be to replace one node by another basic function randomly chosen from the basic function set. Since such a perturbation may have drastic effects, we do not use it.

2. *Leaves.* The leaves are chosen from a terminal set ($x$ or $y$) or from a constant set, which is a continuous interval ($[0, 1]$). We also have to separate the mutation of constants from the mutation of variables, because they are of a different nature. Of course, we could also imagine a mutation process that transforms a constant into a variable and vice versa. However, this seems to be too extreme, except for the case of transforming variables, as we will see.

   (a) *Constants.* Mutation is the only means 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 section 4.5). A constant is replaced by a new value obtained from a uniform random shot within a disk of fixed radius (another parameter of our algorithm) around it (see Figure 11).

   (b) *Variables.* An "internal" mutation (i.e., changing an $x$ to a $y$ or vice versa) is again possible, but we selected a mutation that changes a variable into a randomly chosen constant (see Figure 12). We made this choice on an empirical basis. We noticed

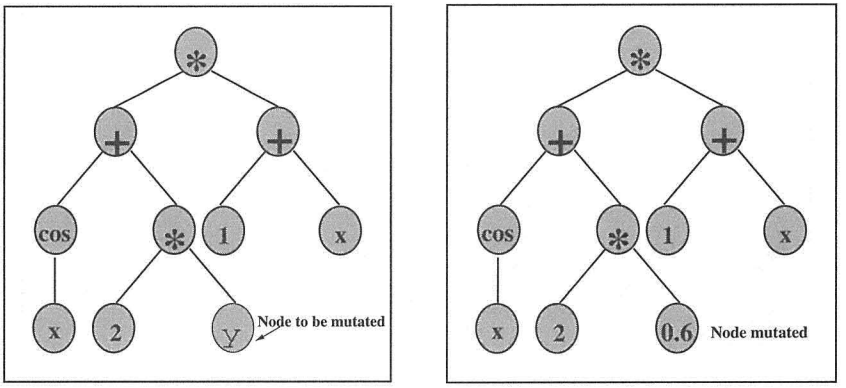


Figure 12: Mutation of variables.

that in some cases constants tend to disappear from the current population. Once they have disappeared, they cannot reappear in the offspring populations. We thus propose the use of a constants creation process, via mutation of variables, to maintain a minimal proportion of constants in the population. The effect of vanishing constants noted experimentally 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 IFS with bad constants too rapidly. This difference is due to the fact that the search space of the nodes and variables is a finite one, whereas the search space of the constants is theoretically infinite. Other techniques (that we have not tested) to avoid the disappearance of constants may be to reduce the size of the constant search space by allowing only a finite set of constants (via sampling, for example) or to separate the symbolic and the numerical optimization (i.e., using a subprocess that optimizes the constants before each IFS evaluation).

### 4.5 Parameter setting

Many parameters have to be tuned to make the algorithm efficient. Here we summarize these parameters and specify practical settings for each.

- *Image size.* The method was tested on images from $64 \times 64$ to $256 \times 256$ pixels.
- *Population size.* Typically 20 to 50 individuals, larger populations were less efficient.
- *Maximum number of generations.* Typically 1000 to 2000. Because small population sizes are used, a large number of generations are

needed for convergence. 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.
- *Mutation 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 number of contractive maps.* From 3 to 7 maps. This is the only constraint 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 during 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 were randomly chosen. We present here three good convergence results. For each example, we specify the target attractor, the best image obtained after convergence, the fitness evolution curve, the parameter setting, and the functions composing the best IFS, compared with the "true" ones (in general, there are an infinite number of IFS leading to the same attractor).

The first point to note is that the functions of the approximations do not resemble those of the target images (especially for Example 1). This is due to the fact that the representation of an attractor by a set of functions is not unique. Parameter adjustment remains a challenging task, but we empirically noticed the following facts.

- The distance images are very efficient. This is particularly obvious from the fitness evolution curves (Figures 14, 16, and especially Figure 18). When updating the distance image, the curve suddenly drops and then rises again. For the new distance image, the value of the fitness becomes lower, because it is computed on a distance image with larger $d_1$ and $d_2$ parameters. This corresponds 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 that yield good results are rather compact; the convergence to line-shaped targets is more difficult.

**Example 1.** Approximation of a square (see Figures 13 and 14 and Table 1 for the parameter settings).

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:

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

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

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

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

Figure 13: Example 1, from left to right: Original image and best images of generations 10, 100, 300, and 1500.

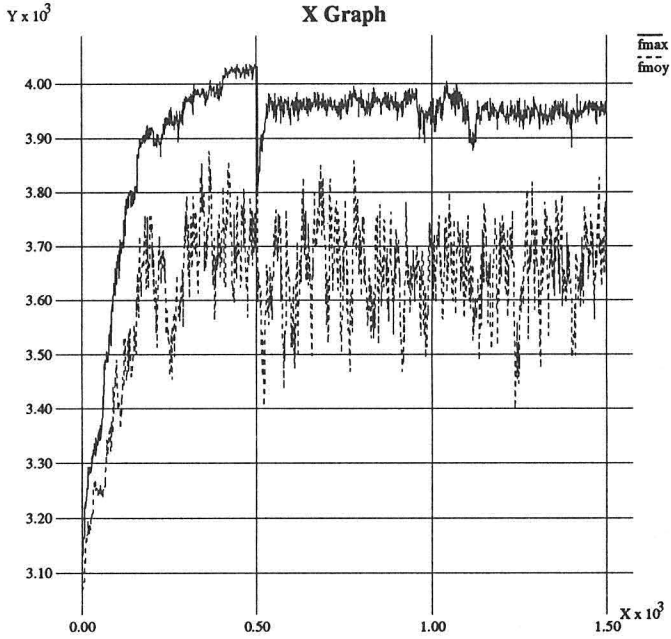


Figure 14: Example 1, fitness evolution. The maximum fitness of the current population is the continuous curve; the mean fitness is the dotted one.

Table 1: Example 1 parameter settings.

| | |
|---|---|
| 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 |

**Example 2.** Approximation of a random IFS (see Figures 15 and 16 and Table 2 for the parameter settings).

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:

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

$$w_2(x,y) = \begin{pmatrix} \log(1+|x|) \\ \cos y \end{pmatrix}$$

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

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

Figure 15: Example 2, from left to right: Original image and best images of generations 50, 300, 500, and 1000.

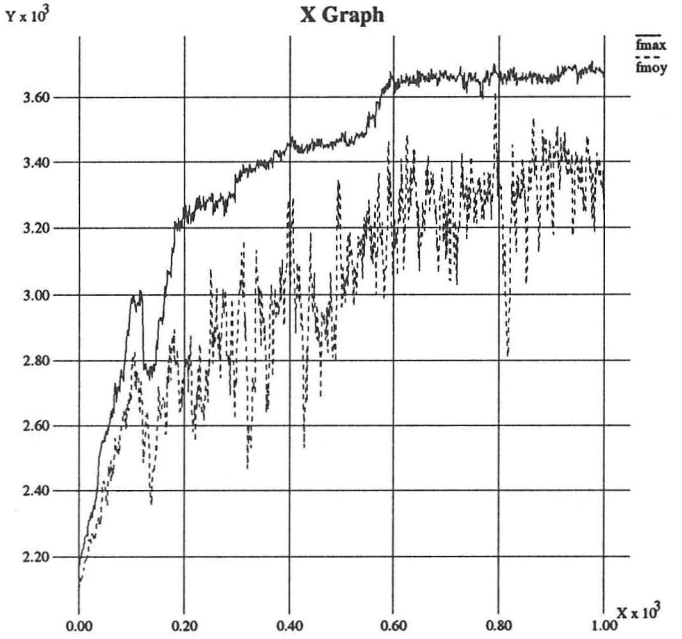


Figure 16: Example 2, fitness evolution. The maximum fitness of the current population is the continuous curve; the mean fitness is the dotted one.

Table 2: Example 2 parameter settings.

| | |
|---|---|
| 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 |

**Example 3.** Approximation of a random IFS (see Figures 17 and 18 and Table 3 for the parameter settings).

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:

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

$$w_2(x,y) = \begin{pmatrix} \log(1+\sqrt{|\cos(\sin x) - \sin(\cos(0.568514 - \cos(\sqrt{|\cos y|})))|}) \\ \sqrt{|\cos(\cos(\sin y) - y * \cos(\cos(\sqrt{\sqrt{y - 0.999847}})))|} \end{pmatrix}$$

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

$$w_4(x,y) = \begin{pmatrix} x * \sqrt{\sqrt{|0.335979 - \cos\sqrt{|x|}|}} \\ \cos y \end{pmatrix}$$

$$w_5(x,y) = \begin{pmatrix} \sin(\sqrt{|\cos y|} + x - \cos(\cos(\sin x))) \\ \cos(\cos(\cos(\cos y))) \end{pmatrix}$$

Figure 17: Example 3, from left to right: Original image and best images of generations 50, 260, 1010, and 1300.

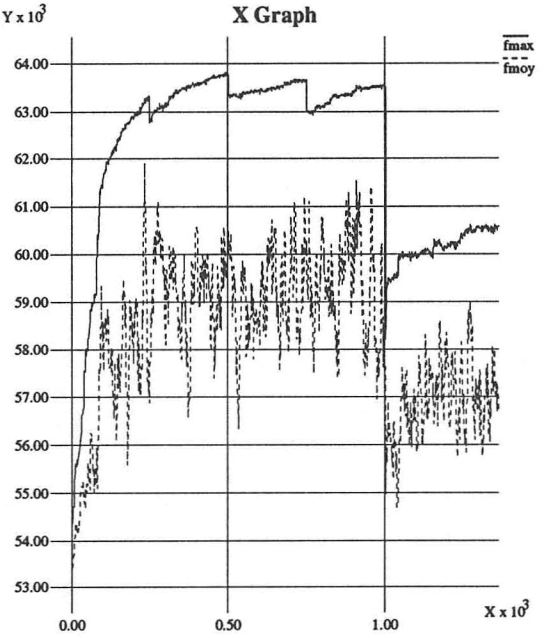


Figure 18: Example 3, fitness evolution. The maximum fitness of the current population is the continuous curve; the mean fitness is the dotted one.

Table 3: Example 3 parameter settings.

| | |
|---|---|
| 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 |

## 6. Conclusion

We have proposed a method to solve the "general" inverse problem for the mixed IFS within a reasonable computation time (a few hours on some modern computers). This computation time is similar to computation times of GA applied to the inverse problem for the affine IFS [18], although in the case of the mixed IFS the size of the search space is much 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 recomputation of distance images at every generation would allow the parameters $d_1$ and $d_2$ to 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 response to the performance of the individual.
- Make the iteration number of the toss-coin evaluation algorithm more adaptive (we can theoretically fix the iteration number and the probabilities of the toss-coin algorithm to more rapidly approximate the attractor within a fixed error).
- Modify the storage structure of the IFS 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 the affine IFS. The use of the mixed IFS may yield more flexible spatial and gray-level transformations, and thus allow the compression ratio to be improved 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 Algorithms with Fuzzy Data," in *Pattern Recognition and Image Analysis*, Selected papers of the 4th Spanish Symposium (September 1990), Perez de la Blanca Ed, pages 97–113, 1992.

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

[3] M. Barnsley, S. Demko, J. Elton, and J. Geronimo, "Invariant Measures for Markov Processes Arising from Iterated Function Systems with Place-dependent Probabilities," (Georgia Institute of Technology, preprint).

[4] M. Barnsley, V. Ervin, D. Hardin, and J. Lancaster, "Solution of an Inverse Problem for Fractals and other Sets," *Proceedings of the National Academy of Science (USA)*, **83** (1986).

[5] M. F. Barnsley, *Fractals Everywhere* (Academic Press, New York, 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 Institute of Technology, 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, Reading, 1989).

[12] D. P. Hardin, "Hyperbolic Iterated Function Systems and Applications," (Georgia Institute of Technology, Ph.D. thesis, 1985).

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

[14] J. Hutchinson, "Fractals and Self-similarity," *Indiana University Journal of Mathematics*, **30** (1981) 713–747.

[15] A. E. Jacquin, "Fractal Image Coding: A Review," *Proceedings of the IEEE*, **81** (1993).

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

[17] J. Livy Vihel, *Analyse et synthèse d'objets bi-dimensionnels par des méthodes stochastiques*, (Université de Paris Sud, Ph.D. thesis, 1988).

[18] E. Lutton and J. Levy-Vehel, "Optimization of Fractal Functions using Genetic Algorithms," (INRIA, research report 1941, June 1993).

[19] E. Lutton and P. Martinez, "A Genetic Algorithm for the Detection of 2d Geometric Primitives in Images," in *12-ICPR* (Jerusalem, Israel, 9-13 October, 1994).

[20] G. Mantica and A. Sloan, "Chaotic Optimization and the Construction of Fractals: Solution of an Inverse Problem," *Complex Systems*, **3** (1989) 37–62.

[21] D. J. Nettleton and R. Garigliano, "Evolutionary Algorithms and a Fractal Inverse Problem," *Biosystems*, **33** (1994) 221–231.

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

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

[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," *Iterated Function Systems: Theory, Applications and the Inverse Problem*, 1991, pages 405–468.

[26] R. Vrscay, "Moment and Collage Methods for the Inverse Problem of Fractal Construction with Iterated Function Systems," in *Fractal 90 Conference*, Lisbonne, June 6-8, 1990.

[27] E. W. Jacobs, Y. Fisher, and R. D. Boss, "Fractal Image Compression using Iterated Transforms," *Data Compression*, 1992.