Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving

Pierre COLLET, Evelyne LUTTON, Frédéric RAYNAL
INRIA - Rocquencourt, B.P. 105, 78153 LE CHESNAY Cedex, France
Pierre.Collet@inria.fr, Evelyne.Lutton@inria.fr, Frederic.Raynal@inria.fr
`http://www-rocq.inria.fr/fractales/`

Marc SCHOENAUER
EEAAX-CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex, France
Marc.Schoenauer@polytechnique.fr
`http://www.eeaax.polytechnique.fr`

May 30, 2000

Abstract

This paper proposes a new method for treating the inverse problem for Iterated Functions Systems (IFS) using Genetic Programming. This method is based on two original aspects. On the fractal side, a new representation of the IFS functions, termed Polar Iterated Functions Systems, is designed, shrinking the search space to mostly contractive functions. Moreover, the Polar representation gives direct access to the fixed points of the functions. On the evolutionary side, a new variant of GP, the "Parisian" approach is presented. The paper explains its similarity to the "Michigan" approach of Classifier Systems: each individual of the population only represents a part of the global solution. The solution to the inverse problem for IFS is then built from a set of individuals. A local contribution to the global fitness of an IFS is carefully defined for each one of its member functions and plays a major role in the fitness of each individual. It is argued here that both proposals result in a large improvement in the algorithms. We observe a drastic cut-down on CPU-time, obtaining good results with small populations in few generations.

1 Introduction

Iterated Functions System (IFS) theory is an important topic in fractals, and provides powerful tools for the investigation of fractal sets (see for example [12, 2, 9]).

The major challenge of both theoretical and practical interest is the resolution of the inverse problem – finding an IFS whose attractor approximates a predefined target [17, 27, 26, 3], be it a 1D curve, a 2D set, or a 3D mesh (image coordinates + grey level or color). A mathematically exact solution can be found only in some particular cases. In general, the problem then turns into an approximation problem which can be solved by optimisation techniques.

The results of these IFS optimisations are used in a wide range of problems amongst which signal analysis (to provide functional representations adapted to signal interpolation [15]), watermarking [20] and image compression [1, 13, 7].

Finding the representation of a complex image (grey-level or color) as an IFS attractor is an extremely hard problem. Therefore mediatic fractal image compression techniques are based on an extremely simplified version of the inverse problem for IFS by setting several strong restrictive hypotheses:

- the functions involved in the IFS are localised on restricted domains and ranges (typically partitions of the image),
- the functions are supposed to be affine.

An approximation of the solution is obtained for each range, using a least square method. As the result is very rough, a way to obtain an acceptable approximation is to simplify the problem by reducing the sizes of the ranges and the domains. The desired compression quality then guides the recursive division process for the ranges and domains (see [7] for precisions). Unfortunately a visually acceptable quality of compression necessitates a huge number of tiny ranges and domains.

Therefore, in the context of image compression, finding a more efficient resolution technique for the general inverse problem for IFS would improve the approximation for each range-domain couple. This means that for an equivalent result, a smaller number of ranges and domains could be used.

A solution using standard genetic algorithms on classical IFS compression schemes (affine functions) has already been proposed in [25].

This paper presents a further step in this direction by relaxing hypotheses (non affine functions) and using advanced EA techniques: improving the quality of IFS approximations can bring significant progress in all applications based on the IFS inverse problem.

1.1 IFS representations

An IFS is a collection of contractive mappings $\Omega = \{F, (w_i)_{i=1,..,N}\}$ (see appendix A for detailed definitions). The main interest of this mathematical object is that it uniquely defines a particular set, called its "attractor" (which is easily built with help of simple iterative algorithms). IFS attractors were originally used to define fractal sets (SIERPINSKI gasket, BARNSLEY fern, etc ...). Many image/signal analysis applications are now based on this technique.

The inverse problem for IFS address here is the following : for a given shape, find a contractive IFS, the attractor of which is as close as possible to this shape. This inverse problem problem can be formulated as an optimisation problem: some computational solutions exist, based on deterministic or stochastic optimisation methods. As the function to be optimised is extremely complex, some *a priori* restrictive hypotheses are necessary. Usually, the search space is that of affine IFS, with a fixed number of functions [1, 14]. Solutions based on Evolutionary Algorithms have recently been presented for affine IFS [26, 8, 25, 19].

Previous work [16] dealt with general non-affine IFS using Genetic Programming (GP), termed *Mixed IFS*:

- such IFS are capable to create a wide variety of shapes,
- GP offers an easy representation for evolving general functions.

However, without any other guideline than target shapes, functions defined by GP parse-trees are rarely contractive. Moreover, their fixed point needs to be numerically estimated.

This paper considers the alternative representation of non-affine IFS. Each function is represented in polar coordinates with respect to some central point. The term "Polar IFS" will be used to designate an IFS built on such functions. There are many advantages to the polar representation:

- a simple constraint on the radial coordinate ensures the convergence towards the central point, which happens to be the fixed point of the function (see section 2),
- polar IFS can be represented as GP parse-trees with a simple wrapper; the associated inverse problem can hence be solved using GP,
- the handling of contractance constraints is simpler than with mixed IFS —although the contractance still has to be checked numerically— as the proportion of contractive IFS in the set of Polar IFS is much larger than in the set of Mixed IFS (see section 2.2). Hence, polar IFS provide a more efficient (less sparse) search space to the optimisation algorithm than Mixed IFS.

1.2 IFS evolution

All of the above-mentioned works dealing with the inverse problem have used what can be called a "standard" approach to evolve IFS: an individual is a fully fledged IFS, made of several functions —independent of the representation (polar or mixed). In this approach, the solution is the best individual of the final population. All individuals but one are discarded, which seems a great waste, as the elaboration of each of them has used the same amount of CPU-time as the winner.

Alternative approaches have been proposed in the Classifier framework: in Classifier Systems (CS), the "Pittsburgh" approach [5, 24] individuals are complete rule bases, whereas in the "Michigan" approach [11, 10, 29] individuals are single rules, and the solution is built using several individuals (rules) of the population.

In this paper, the "Michigan" approach is transposed into the framework of IFS, and termed *Parisian approach* (section 3): single contractive functions are evolved using GP, and IFS are built using individuals (functions) from the population. The main difficulty of this approach is the definition of the partial fitness for a single function —the direct transposition of the Bucket Brigade algorithm used in CS is impossible. Section 4 proposes a problem-specific approach.

Finally, section 5 describes how *Polar IFS* and the *Parisian approach* are implemented to solve some instances of IFS inverse problem. Results on three images, together with comparisons with previous results from [16] are presented and acknowledge the power of the proposed approach.

2 Polar IFS

This section introduces the Polar representation of non-affine IFS, and discusses the contractance issue. Experiments are presented to show the advantage of Polar IFS *versus* general non-affine IFS.

2.1 Polar representation

The main difficulty which arises when manipulating non-linear IFS is the handling of the contractance constraint. There is no general analytic way to check the contractance of a non-affine function, and experimental tests require heavy computations while only giving an idea on the contractance of the function. For instance, less that 15% of random GP trees actually are contractive (see [16], and also experiments of section 2.2). This empirical fact motivated the introduction of an alternative representation.

2.1.1 Local contractance

The first idea is to define a weaker contractance condition than the usual global condition (see Definition 2 in Appendix A), that will be easier to check:
Definition: A mapping $w : F \to F$, from a metric space (F, d) into itself, is called *locally contractive with respect to point* P if there exists a positive real number $s < 1$ such that:

$$\forall M \in F,^2 \quad ||\overrightarrow{P\ w(M)}|| < s||\overrightarrow{PM}||$$

The smallest of such numbers s is called the *local contraction ratio* of w.

It is obvious that if w is locally contractive w.r.t. P, then P is the unique fixed point of w. Nevertheless, local contractance does not imply global contractance, as demonstrated by the counter-example of appendix B. Though this counter-example seems to ruin all efforts to easily design contractive mappings to use to build IFS, the next sections will show that indeed, benefits can arise from using locally contractive functions in IFS.

2.1.2 GP representation and wrapper

When using GP trees to represent IFS, two approaches are possible to tackle the contractance requirement: wrap GP trees into contractive mappings, or eliminate *a posteriori* non contractive mappings. The latter approach always works, but can be very time consuming, as many trees that are generated along the evolution need to be eliminated (see [16]). This eugenistic approach was proposed because no cleaner way could be found to generate contractive mappings in two dimensions.

But, together with local contractance goes the idea of polar representation: if one considers a locally contractive mapping w with a fixed point P, then points can be represented using (ρ, θ) coordinates centered on P,[1] and w itself can thence be defined easily using such coordinates. The important fact is that the local contractance condition becomes a contractance condition on the one-dimensional function giving ρ —and it is straightforward to transform a random one-dimensional function into a contractive function.
Definition: A *Polar IFS* is a set of locally contractive functions w_i with respective fixed points P_i such that each w_i is defined in polar coordinates w.r.t. P_i by two one-dimensional functions F_i and G_i by:

$$w_i \begin{pmatrix} \rho \\ \theta \end{pmatrix}_{P_i} = \begin{pmatrix} \rho \dfrac{th(k_i * F_i(\rho, \theta)) + 1}{2} \\ G_i(\rho, \theta) \end{pmatrix}_{P_i} \tag{1}$$

for some real number k_i.

[1] i.e.: $\overrightarrow{PM} = (\rho\cos(\theta), \rho\sin(\theta))$

The particular form of the ρ function ensures the local contractance —as $\dfrac{th(k_i * F_i(\rho,\theta)) + 1}{2} < 1$ whatever the value of F_i (see figure 2).

A locally contractive function can hence be represented by one point P in $[0,1]^2$ and two functions F and G, which can be evolved as GP trees. Equations (1) are used as a wrapper, which ensures the local contractance of the mapping. Moreover, figure 1 shows that a wide variety of shapes can be obtained using that representation.

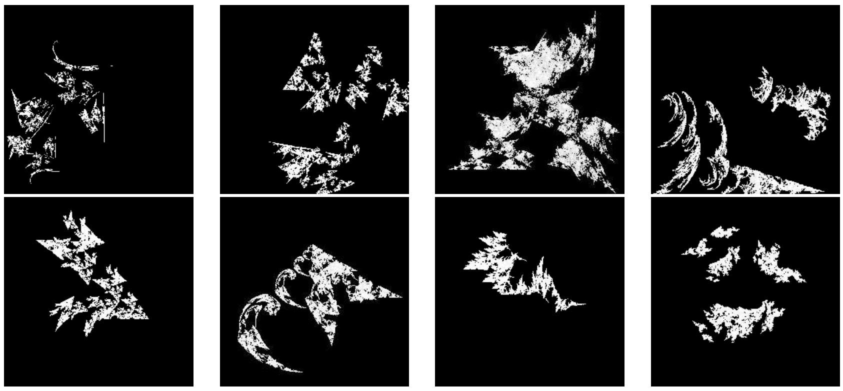


Figure 1: Examples of Polar IFS attractors

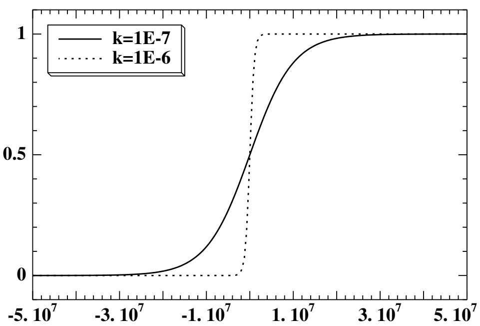


Figure 2: $y = \dfrac{th(kx)+1}{2}$ curve with $k = 10^{-7}$ and $k = 10^{-6}$

2.2 Contractance tests

The counter-example given in Appendix B highlights the fact that contractance verification cannot be avoided for Polar IFS. However, this section will experimentally show that the proportion of contractive mappings is larger in the set of locally contractive mappings defined by equation (1) than in the set of general Mixed IFS. Hence the cost of contractance verification will be much lower for Polar IFS than for Mixed IFS.

This section presents some statistical tests that have been performed on two different sets of functions in order to experimentally approximate their respective proportion of contractive functions. The first set of functions is the set of mixed IFS, i.e.: general non-linear IFS built from general random GP trees. The second set is the set of locally contractive functions defined in section 2.1.1, whose components F and G are random GP trees. In both cases, the random trees are built from the basic nodes and terminals described in section 4.1.

As an analytical computation of the contraction ratio is generally out of reach, a numerical estimation needs to be done experimentally: for each mapping w, a sample of pairs of distinct points (P_i, Q_i) in $[0,1]^2$ is defined, and the minimum of $\dfrac{d(w(P_i), w(Q_i))}{d(P_i, Q_i)}$ gives the approximate contraction ratio for w.

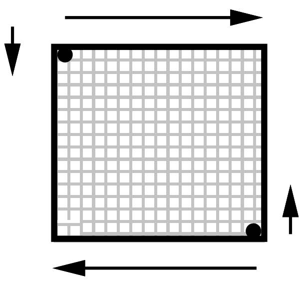


Figure 3: The Twist test.

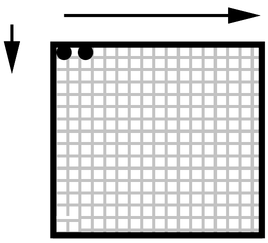


Figure 4: The Square test.

The cost of the approximation is of course proportional to the number of points in the sample: four tests with different samples have been tried on $n \times n$ images, and the results are presented in Tables 1, 2 and 3. Three tests are based on a pre-defined set of pairs (depending on the resolution of the image), while the fourth one selects points uniformly.

1. **All-pixels test**:
   Every pixel of the image is checked with every other pixel of the image. It is the most accurate test, but also the most CPU-consuming.

2. **Twist test**:
   This test starts from the two pixels at two opposite corners of the image, and each point scans symmetrically each half image, line by line, until the center pixel is reached (see figure 3).

3. **Square test**:
   All pixels of the image are tested against their immediate neighbour (see figure 4).

4. **Random test**:
   n pairs of points are selected randomly, uniformly in $[0, 1]^2$.

For some yet unclear reasons[2], the Twist test seems quite inefficient, while the Square test seems much more accurate (i.e.: gives results closer to the All-pixels test) on Polar IFS than on mixed IFS. Hence the Square was found to realize a good compromise between efficiency and cost, and was used for all experiments presented in the following.

Tables 2 and 3 show two other major results:

1. The Square test results are not very far from the All-pixels test results for Polar IFS. Furthermore, the ratio of functions deemed contractive does not decrease drastically with the resolution of the images: less than 2.55% difference between 128×128 and 512×512. This means that the Square test can be considered as quite accurate, regardless of the image resolution above 128×128.

2. the proportion of contractive Polar IFS much larger than that of Mixed IFS: more than 50% compared to less than 10%.

3 Parisian approach in evolutionary computation

The standard approach when using evolutionary methods as stochastic optimisers is to evolve a population of potential solutions to the problem at hand, each of them called individuals. The output of the algorithm is in that case the best individual encountered during the evolution. All other individuals are discarded, whatever information they might bear.

In some cases, however, the search space is a space of lists of items, and an alternate possibility is to evolve a population of such items, and to build a solution by combining different items of the current population. Such an approach is very popular in Classifier Systems: The well-known Michigan approach evolves

[2] A simple explanation might be that the contactance condition is more likely violated by pairs of points that are close from each other.

	# of w_i computed	# of comparisons
all pixels	n^2	$\frac{(n^2-1)(n^2-2)}{2}$
twist	n^2	$\frac{n^2}{2}$
square	n^2	n^2
random	$2n$	n

Table 1: Complexity of the various tests

n	# of w_i computed	all-pixels	twist	square	random
16	256	10.26	24.46	21.64	14.67
32	1024	10.46	24.16	21.80	13.73
64	4096	9.42	23.96	21.62	13.25
128	16384	-	24.03	22.08	13.39
256	65536	-	23.94	21.19	13.04
512	262144	-	23.91	21.18	-

Table 2: Proportion of contractive functions (%) for each test for *mixed* IFS

n	# of w_i computed	all-pixels	twist	square	random
16	256	62.48	90.56	64.94	74.29
32	1024	58.82	88.91	60.72	69.28
64	4096	56.12	88.08	57.91	64.30
128	16384	-	87.39	56.44	62.14
256	65536	-	87.24	55.26	60.00
512	262144	-	87.03	53.89	-

Table 3: Proportion of contractive functions (%) for each test for *polar* IFS

single rules, and a solution is a rule base made of some of the best individuals of the final population [11, 10, 29]. On the other hand, the so-called Pittsburgh approach evolves populations of complete rule bases [5, 24].

Similarly to the Michigan approach in CS, the *Parisian approach* proposed in this paper consists in evolving a population of locally contractive functions, and to build an IFS by picking up functions in the population.

Clearly, only problems where the solution can be set apart into separate components can be handled using a Parisian approach. However, another necessary condition is to be able to accurately evaluate the *local fitness* of a single component, i.e.: its usefulness to the global fitness of the solution.

Fortunately, the IFS inverse problem offers some nice ways to evaluate the contribution of a single function of the system to its global performance: consider the target shape A; if an IFS (w_i) is a solution of the inverse problem, then $A = \cup w_i(A)$. Hence a local fitness for a w_i should consider positively the part of $w_i(A)$ lying inside A, and negatively the part of $w_i(A)$ lying outside A. Furthermore, the position of the fixed points also gives some indications. It should lie inside the target shape A, but the relative positions of the fixed points of two locally contractive functions also define semi-distances between individuals, which can be used to implement some sharing mechanism to keep diversity in the populations and prevent all functions in the populations to become similar.

4 Polar IFS + Parisian GP

4.1 GP components

As described in section 2.1, a locally contractive mapping is defined by two functions ($F(\rho,\theta)$ and $G(\rho,\theta)$) and a point P. Whereas P is simply represented by its two real-valued coordinates that do not need further description, F and G are modeled as GP-trees, similarly to Mixed IFS [16] .

Nodes and terminals: GP trees are built from a set of terminals, which consist of the variables ρ and θ, constants in $[-1,1]$, and a set of nodes, built from the following set of basic real-valued functions.

Unary nodes:[3]

- $-x$
- $\frac{1}{x}$ for $x \notin [-1, 1]$
- $\cos(x)$
- $\sin(x)$
- $th^{-1}(x)$
- $psqrt(x) = sign(x)\sqrt{|x|}$
- $plog(x) = sign(x)\log(|x|)$, if $x \neq 0$

Binary nodes:

- $+$
- $-$
- $\times$
- $\div$

Initialisation: The initialisation procedure for the GP trees is the so-called *grow* procedure, consisting in a simple recursive random choice into the joint set of node and terminals until either terminals are drawn on all branches or the maximum depth is reached.

As for the fixed points, they are randomly chosen among the contour points of the target shape A. This idea comes from the conjecture by Dekking [6] that there always exists solutions to the inverse problem where fixed points are on the edges of the target shape. This result has been proven in the case of affine IFS in [6].

Crossover: the standard GP crossover is used on both trees: it performs swaps of randomly selected sub-trees between the parents. The fixed points are not modified by crossover.

Mutation : Different mutation operators are used, taking into account both the trees representing the F and G functions and P, the center of the locally contractive mapping at hand.

- No mutation acting on nodes has been retained, due to the too drastic effects of such a perturbation on the phenotypic behavior of the corresponding tree.
- For the same reason (too drastic effect), the only mutations of terminals finally retained are the mutation of the values of constant terminals and the mutation of variable terminals into constant terminals.
- *Constants*: the precise adjustment of constant terminals is critical when handling numerical trees (i.e.: trees that represent a real-valued function).

[3] $sign(x)$ returns the sign of x, i.e.: $sign(x) = 1$ if $x \geq 0$ $sign(x) = -1$ otherwise.

Whereas John KOZA's seminal work considered only a finite number of possible values for constant terminal, relying on arithmetics to possibly generate any value (e.g. using $+$, * and $\div$ it is possible to approximate within a given precision any real value from integers in $[0,1]$), more recent works consider that specific mutation operators are a better approach to the fine tuning of constant terminal values. In this perspective, constant terminal values are modified here by choosing a new value uniformly from a disk of fixed radius (another user-defined parameter).

Early experimental investigations highlighted the fact that constant terminals tend to disappear from the population. One possible explanation is that the numerical optimisation of the constant values is a more difficult task than the symbolic optimisation of the other nodes. The selection operator thus tends to rapidly eliminate functions having wrong constant values. This difference could come from the fact that the search spaces of the nodes and variables are finite while the search space of the constants is infinite. One way to tackle this difficulty is to implement an optimisation loop for constant values after the application of any variation operator (mutation or crossover). Though quite successful in other frameworks [23], this technique still needs to be tested in the IFS framework.

- *Variables*: mutations of variables considered here is either their transformation into constant terminals, the value of which is randomly drawn within $[0,1]$, or their transformation into another variable.
- *Fixed points*: the mutation operator moves the fixed point of a contractive function by choosing another point uniformly in a neighborhood of radius 4 pixels from the parent fixed point, while staying on the boundary of the target shape.

A niching strategy is mandatory to prevent all individuals to rapidly become similar to the best covering mapping. Here, the dynamic niche sharing has been used [18]. This technique is based on a clusterisation of the populations with respect to a predefined inter-individuals distance. The selection operator is then adapted in order to fairly select a given percentage of best individual of each niche.

Table 4.1 summarizes the parameters used for the evolution. The implementation is based on the GALib library [28].

4.2 Fitness function for individuals

The fitness of an individual in the Parisian approach is divided into two main parts: the *local* fitness, computed from the characteristics of the individual itself, and the *global* fitness, that is based on a measurement of how a set of individuals actually solves the problem at hand. In the IFS inverse problem framework, the

BINARY NODES	$+$, -, *, $\div$	
UNARY NODES	**neg, 1/., cos, sin, th^{-1}, psqrt, plog**	
POPULATION SIZE	see results, section 5	
OPERATORS	**Mutation probabilities**	
	constant $\rightarrow$ constant	0.15 according to a Gaussian law of variance 0.1
	variable $\rightarrow$ constant	0.05 randomly chosen in $[-1, 1]$
	constant $\rightarrow$ variable	0.08
	variable $\rightarrow$ variable	0.08
	function $\rightarrow$ function (same arity)	0.08
	fixed points:	0.1, linearly decreasing along the generations.
	Crossover probability	
	PCROSS	0.95 for trees no crossover for fixed points
SELECTION	**Ranking**	
	selection pressure	1.35
REPLACEMENT	**Population replacement scheme**	
	replacement percentage	50% Overlapping populations
STOPPING CRITERION	**Based on approximation of target**	
		see section 4.3
WRAPPER	**Polar description of contractive mappings**	
		see section 2
SPECIAL FEATURES	**Dynamic niche sharing**	
	σ	0.05
	Max nb of clusters	0.5 of POP_SIZE

Table 4: Parameters for the evolution of polar IFS

local fitness takes into account the relative positions of the attractor of a single contractive mapping and the target shape.

4.2.1 First goal: $w_i(A)$ must lie inside A

Let $\#[A]$ be the number of pixels of $A \subset F$, and let us define $\mathcal{F}_1(w_i)$ as:

$$\mathcal{F}_1(w_i) = \frac{\#[w_i(A) \bigcap A]}{\#[w_i(A) \bigcap A] + \#[w_i(A) \backslash A]}$$

$\mathcal{F}_1(w_i)$ is maximum (and is equal to 1) iff $w_i(A) \subset A$.

4.2.2 Second goal: maximize the area of $w_i(A) \bigcap A$

Similarly, define $\mathcal{F}_2$ to account for the maximisation of the size of $w_i(A) \bigcap A$:

$$\mathcal{F}_2(w_i) = \frac{\#[w_i(A) \bigcap A]}{\#[A]}$$

$\mathcal{F}_2(w_i)$ is maximum (and is equal to 1) iff $A \subset w_i(A)$.

4.2.3 Integration of the contractance constraints

As described in section 2.2, the contractance test can be included in the computation of the image of the target $w_i(A)$. At the same time, the mean contraction ratio s_i can be estimated. If the function is not contractive, $\mathcal{F}_1$ and $\mathcal{F}_2$ defined above are not computed and are directly set to zero.

4.2.4 Local fitness

The local fitness is defined as a linear combination of $\mathcal{F}_1$, $\mathcal{F}_2$ and the distance to 1 of the estimated contraction ratio:

$$\mathcal{F}_{loc}(w_i) = \mathcal{F}_1(w_i) + \mathcal{F}_2(w_i) + (1 - s_i)$$

where s_i is the estimated contraction ratio of w_i. Due to the term $\mathcal{F}_2$, $w_i(A)$ tend to fill A, which is not satisfactory for an IFS. This effect is counterbalanced by the term $1 - s_i$ which forbids the trivial solution $w_i = Id$.

This fitness represents an interpretation of the collage property of an IFS, i.e.: one searches for the set of best w_i's such that $A = \bigcup w_i(A)$. This is yet another argument for the use of some niching mechanism in order to avoid that all individuals go to the same best coverage of the target shape. Moreover, a side effect might be an "economic" coverage of A, i.e. with the smallest possible $w_i(A) \bigcap w_j(A)$ for all i, j.

4.3 Global fitness and its repartition on individuals

A clusterisation of the current population with respect to a distance defined on the search space (the mean distance of the images of a set of sample points) yields the set of the N individuals to be globally evaluated. These N best individuals build an IFS Ω which represents a potential solution to the inverse problem at hand. A toss-coin algorithm is used in order to compute its attractor A_Ω, which is then compared to the target A using two quantities:

$$\begin{array}{lll} In_\Omega & = \frac{\#[A_\Omega \bigcap A]}{\#[A]} & \text{proportion of points of } A_\Omega \text{ within the target} \\ Out_\Omega & = \#[A_\Omega \backslash A] & \text{number of points of } A_\Omega \text{ without} \end{array}$$

The global fitness at generation n takes into account both the attractor of the current IFS $\Omega(n)$ and the attractor of the IFS constructed at generation $n-1$, $\Omega(n-1)$. A *global parsimony* term is added to favour solutions with a smaller number of functions (nothing to do with usual GP parsimony that takes into account the size of the trees).

$$\begin{aligned} F_{glob}(n) \quad = \quad & [In_{\Omega(n)} - In_{\Omega(n-1)}] - [Out_{\Omega(n)} - Out_{\Omega(n-1)}] \\ & +\alpha[\text{Nb\_functions}(\Omega(n)) - \text{Nb\_functions}(\Omega(n-1))] \end{aligned}$$

For all results presented in section 5, the parsimony factor α was set to 0.075.

This global fitness is simply added to the individual fitness of the "active" individuals w_i of the population, according to their participation in the current IFS:

- if w_i just entered the IFS $\Omega(n)$ (it did not participate in $\Omega(n-1)$) then:

$$Fitness(w_i) = F_{loc}(w_i) + F_{glob}(n)$$

- if w_i was already present in $\Omega(n-1)$:

$$Fitness(w_i) = F_{loc}(w_i) + \frac{F_{glob}(n) + F_{glob}(n-1)}{2} \times \frac{1}{[age(w_i)]^2}$$

  where $age(w_i)$ stands for the number of generations during which w_i has been part of the IFS.

- if w_i did just quit the IFS:

$$Fitness(w_i) = F_{loc}(w_i) - F_{glob}(n)$$

- if w_i does not belong to the IFS:

$$Fitness(w_i) = F_{loc}(w_i) + \frac{F_{glob}(n-1)}{2}$$

  (The term corresponding to the global fitness decreases along generations.)

In other words, the global contribution term distributed on each individual takes into account the past of this individual and its age with respect to the current IFS.

Termination criterion: F_{glob} is also used as a stopping criterion: the algorithm stops whenever the target is approximated within a fixed threshold, based on the F_{glob} value.

4.4 Improving the toss-coin algorithm

The usual stochastic method known as *toss-coin* has been chosen here to actually compute the attractors, as it is acknowledged to be less CPU-consuming than the deterministic algorithm (see section A.2). Nevertheless, the computation of the attractors is by far the most costly step of the whole inverse problem solving process. Hence, great care must be given to its actual implementation. The notion of *patience* has been introduced to cut down the computation time of unpromising evaluations of attractors with the toss-coin algorithm.

All calculations are done in the $[0,1] \times [0,1]$ unit square, discretised into a finite image (e.g. 512×512). The following possibilities might then happen:

1. the attractor is almost uniformly spread within the target shape,
2. the attractor is almost uniformly spread across the 512×512 image,
3. the attractor lies mostly out of the target shape,
4. the attractor lies within a very small (e.g. 2×2 pixels) area.

IFS attractors have an incredible variety of shapes. Hence it is unfortunately very unlikely that they should produce attractors of the first or second kind in the first generations.

Moreover, the number of pixels is finite. This means that the drawing speed (frequency of apparition of *new* pixels in the attractor) decreases along the iterations.

Adaptive optimisation criterion Deciding once for all that the toss-coin routine should be iterated 10,000 times for instance is a very bad decision: if the attractor is of the third or fourth kind, a lot of cpu time is lost. On the other hand, 10,000 iterations may not be enough to draw a faithful representation of the attractor in the first two cases.

Optimizing the number of iterations is crucial. Ideally, one should stop iterating the toss-coin function as soon as the drawing speed of the attractor comes close to 0.

The notion of *patience* is introduced to adaptively adjust the number of iterations: suppose a patience of 1,000 iterations. If no newer pixel has been illuminated on the discretised image during the last 1,000 iterations, the algorithm stops – guessing that no significant amount of new pixels will come out of the toss-coin routine in the future.

Formally speaking, the value of the `Patience` parameter sets a threshold on the speed of occurrence of new pixels on the image. Moreover, the patience criterion automatically *adapts* to both the definition of the final image, and the required precision:

- On a reduced 64×64 image, each pixel represents a large subset of the $[0,1] \times [0,1]$ domain, and the toss-coin stops rapidly.
- During the first generations, a very fine representation of the attractor is superfluous: one only needs to roughly know if the IFS lies out of the target shape or not. One can then mischievously increase the value of the `Patience` variable along with the generations: in all experiments, the `Patience` is initially set to 50 and is incremented with every generation. A patience of 1,000 is used to produce the final image.

5 Results

5.1 Sample results

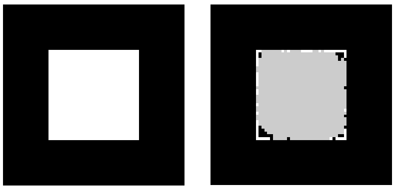
Figures 5 to 8 present some results of the proposed algorithm for some inverse IFS problems. The parameter settings are those of table 4.1.

5.2 Comparative results

Comparison with previous approaches of the inverse problem for IFS is rather difficult. Indeed, current improvements have come from two sources:

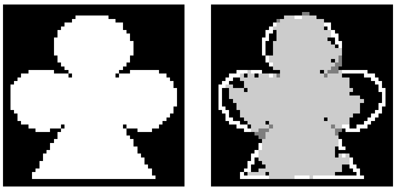
- the use of Polar IFS (i.e. a restriction of the search space) has provided an easier search space for the optimisation algorithm,
- the Parisian strategy for GP is a very particular way of handling the evolutionary optimisation. Comparisons in terms of function evaluation with the classical approach is not very meaningful, as an evaluation in a classical evolutionary algorithm and in the Parisian approach do not represent the same thing with respect to the function to be optimised.

Figures 9 and 11 present results obtained on the square target. The most reasonable comparison criterion is the number of GP tree evaluations. This is roughly equivalent to *Population Size* $\times$ *Number of Generations/2 + Number of Selected Individuals for a Global evaluation* $\times$ *Number of Generations* for the



Target 49 generations
14 functions
pixels inside: 85% (light gray)
pixels outside: 0% (dark gray)

Figure 5: 64x64 target, with a population size of 60 individuals.

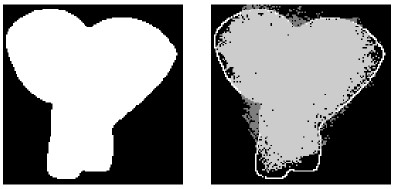


Target 60 generations
13 functions
pixels inside: 70% (light gray)
pixels outside: 2.2% (dark gray)

Figure 6: 50x50 target, with a population size of 100 individuals.

Target 400 generations
13 functions
pixels inside: 60% (light gray)
pixels outside: 0.36% (dark gray)

Figure 7: 80x80 Dolphin target, with a population size of 100 individuals.



Target

87 generations
31 functions
pixels inside: 84.52%
pixels outside: 5.03%

Figure 8: 128x128 target, with a population size of 200 individuals.

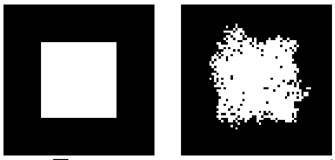
Parisian approach (only 50% of the population is replaced at each generation, hence the "*/2*"), compared with *Number of Individuals* $\times$ *Number of Generations* for the classical approach.

- Figure 9 presents results obtained with a Genetic Algorithm for affine IFS (searching for a 4 functions IFS, i.e. for 24 real parameters) this result was obtained using approximatively 200,000 evaluations (a population size of 20 individuals during 10,000 generations). Figure 10 presents the result of an iterative implementation of the same algorithm where successive runs are made on more and more precise approximations of the target (best individuals of the previous run are included in the initial population of the next GA run). Fewer iterations were necessary in order to obtain similar results (around 55,000).
- Figure 11 presents results obtained with a Genetic Programming technique for mixed IFS (from [16]). These results were obtained using approximatively 45,000 evaluations (population size of 30 during 1,500 generations).
- Using an individual scheme on Polar IFS, only 1,519 evaluations were necessary to obtain results of figure 5: 60 (individuals) $\times$ 49 (generations) / 2 (50% replacement) + 49 (global evaluations).

6 Conclusion and perspectives

This paper has introduced two original features in the framework of IFS inverse problem:

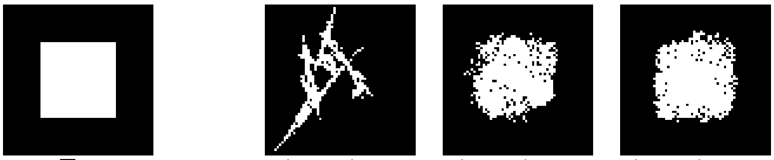
- on the fractal side, Polar IFS are an interesting model which simplifies the manipulation of non-linear IFS. Moreover, the very high proportion of contractive mappings in the set of locally contractive mappings, together



Target

10000 generations
4 functions
pixels inside: 88.4%
pixels outside: 17%

Figure 9: Classical GA for affine IFS on a 64x64 target, with a population size of 20 individuals.



Target	iteration 1	iteration 2	iteration 3
	500 gen.	1,500 gen.	2,500 gen.
	5 indiv.	10 indiv.	15 indiv.
	inside: 31%	inside: 79%	inside: 88%
	outside: 9%	outside: 10%	outside: 7%

Figure 10: Classical GA for affine IFS: iterative scheme for a 64x64 square (4 functions).

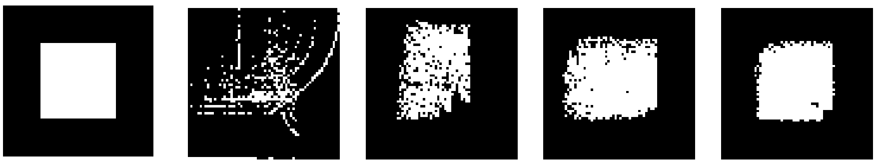


Figure 11: Genetic Programming for Mixed IFS on a 64x64 target. From left to right: target and best images of generations 10, 100, 300 and 1,500. Population size 30 individuals.

with the simple wrapper transforming two general GP trees into a locally contractive mapping, results in a very efficient search;

- on the evolutionary side, the IFS framework is an area where a Parisian approach is possible: single functions are evolved, and a subset of the population actually builds the solution to the inverse problem. The careful design of both the local and the global fitness functions, as well as their balanced aggregation into the final fitness used for selection, are crucial for the success of the method.

These first experiments indicate that making use of *a priori* information on the problem to solve can be quite efficient with the Parisian approach. Moreover an approximation of the target shape is obtained very rapidly while the representation of details needs more computation time.

Future work on this topic concern:

- cross-validation tests using the Parisian approach with mixed IFS representations, and the standard approach with the Polar IFS representation, in order to sort out the respective benefits of both original improvement introduced here;
- implementation of a Parisian GP technique in an interactive manner for artistic generation of fractal images;
- use of the Polar IFS representation as an alternative representation for mechanical structures, in the field of topological optimum design [22].

Acknowledgements

The authors would like to thank Jean-Pierre TILLICH for his help in the construction of the counter example of section 2, Jacques LÉVY VÉHEL for the numerous discussions we had about inverse problem for IFS, and Laurent BALAGUÉ who helped us a lot in the final implementation of the techniques described in this paper.

References

[1] M. F. Barnsley, *Fractals Everywhere*, Academic Press, New-York, 1988.

[2] M. Barnsley and S. Demko, Iterated Function System and the Global Construction of Fractals, *Proc. of the Royal Society*, A 399:243–245, 1985.

[3] 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.

[4] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, Individual GP: an Alternative Viewpoint for the Resolution of Complex Problems, in W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, R. Smith (Eds.), *Genetic and Evolutionary Computation Conference*, Morgan Kaufman, pp. 974–981, 1999.

[5] K.A. De Jong, The Analysis of the Behavior of a Class of Genetic Adaptive Systems, PhD thesis, University of Michigan Press, Ann Harbor, 1975.

[6] F.M. Dekking, Fractal Image Coding: Some Mathematical Remarks on its Limits and its Prospects, chapter in Yuval Fisher (Ed.), *Fractal Image Encoding and Analysis*, NATO ASI Series F: Computer and System Sciences, Vol. 159, pp. 117–133, Springer Verlag, 1998.

[7] Y. Fisher, *Fractal Image Compression: Theory and Application*, Springer Verlag, New-York, 1995.

[8] B. Goertzel, H. Miyamoto, Y. Awata, Fractal Image Compression with the Genetic Algorithm, *Complexity International*, 1:25–28, 1994, http://www.csu.edu.au/ci/vol1/goertzel.html.

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

[10] J.H. Holland, Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-Based Systems, in R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, vol.2, pp. 593–623, Morgan Kauffman, 1986.

[11] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Harbor, 1975.

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

[13] A.E. Jacquin, Fractal Image Coding: A Review, *Proceedings of the IEEE*, 81(10):1451–1465, 1993.

[14] J. Lévy Véhel, Analyse et synthèse d'objets bi-dimensionnels par des méthodes stochastiques, PhD thesis, Univ. de Paris Sud, Décembre 1988.

[15] J. Lévy Véhel, K. Daoudi, and E. Lutton, Fractal Modeling of Speech Signals, *Fractals*, 2(3):379–382, 1994.

[16] E. Lutton, J. Lévy Véhel, G. Cretin, P. Glevarec, and C. Roll, Mixed IFS: Resolution of the Inverse Problem Using Genetic Programming, *Complex Systems*, 9:375–398, 1995.

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

[18] B. L. Miller and M. J. Shaw, Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization, IlliGAL Technical Report 95010, 1995.

[19] D. J. Nettleton and R. Garigliano, Evolutionary Algorithms and a Fractal Inverse Problem, *Biosystems*, 33:221–231, 1994.

[20] J. Puate and F. Jordan, Using Fractal Compression Scheme to Embed a Digital Signature in an Image, in T Chiueh, A G Tescher (Eds.), *Video Techniques and Software for Full-Service Networks*, volume 2915 of SPIE Proceedings, pp. 108–118, 1996.

[21] F. Raynal, E. Lutton, P. Collet, and M. Schoenauer, Manipulation of Non-Linear IFS Attractors Using Genetic Programming, in P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala (Eds.), *Congress on Evolutionary Computation 99*, pp. 1171–1177, IEEE Press, 1999.

[22] M. Schoenauer, Representations for Evolutionary Optimization and Identification in Structural Mechanics, in J. Périaux, G. Winter (Eds.), *Genetic Algorithms in Engineering and Computer Sciences*, pp. 443–464, John Wiley, 1995.

[23] M. Schoenauer, M. Sebag, F. Jouve, B. Lamy, and H. Maitournam, Evolutionary Identification of Macro-Mechanical Models, in P. J. Angeline and Jr K. E. Kinnear (Eds.), *Advances in Genetic Programming II*, pp. 467–488, MIT Press, 1996.

[24] S. F. Smith, Flexible Learning of Problem Solving Heuristics Through Adaptive Search, in A. Bundy (Ed.), *Proceedings of IJCAI83*, pp. 422–425, Morgan Kaufmann, 1983.

[25] L. Vences and I. Rudomin, Genetic Algorithms for Fractal Image and Image Sequence Compression, in *Proceedings Computacion Visual 1997*, pp. 35–44, Universidad Nacional Autonoma de Mexico, 1997, http://sgio2.cem.itesm.mx/rudomin/VISUALPS/lucy.pdf.

[26] E. R. Vrscay, Fractal Geometry and Analysis, The Mandelbrot Festschrift, Curacao 1995, in C. J. G. Evertsz, H.-O. Peitgen, R. F. Voss (Eds.), *Iterated Function Systems: Theory, Applications and the Inverse Problem*, pp. 405–468, World Scientific, 1996.

[27] E. R. Vrscay, Moment and Collage Methods for the Inverse Problem of Fractal Construction with Iterated Function Systems, in H.-O. Peitgen, J. M. Henriques, L. F. Penedo (Eds.), *Fractal 90 Conference, Fractals in Fundamental and Applied Sciences*, pp. 271–289, North Holland, 1991.

[28] M. Wall, GALib: A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/ga/

[29] S. W. Wilson and D. E. Goldberg, A Critical Review of Classifier Systems, In J. D. Schaffer (Ed.), *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pp. 244–255, Morgan Kauffman, 1989.

A Iterated Function Systems

This appendix briefly recalls the basis of IFS theory and the numerical algorithms most widely used to compute the attractors of an IFS.

A.1 Notations and definitions

Definition 1: Let (F, d) be a complete metric space, and $(w_i)_{i=1,..,N}$ be a collection of functions defined from F into F. $\Omega = \{F, (w_i)_{i=1,..,N}\}$ is called an *IFS (Iterated Function System)*.

A central notion in IFS theory is the notion of contractive mapping:
Definition 2: 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:

$$\forall (x, y) \in F_{;}^{2} \quad d\big(w(x), w(y)\big) \le s.d(x, y)$$

The smallest of such numbers s is called the *contraction ratio* of w.

A crucial result about contractive mappings is the following:

Theorem *Contractive Mapping Fixed Point Theorem:*
If (F, d) is a complete metric space, and $W : F \to F$ is a contractive mapping, then W has a unique fixed point.

All mappings can also be applied to subsets of F, and give the following:
Definition 3: An IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$ induces an operator W defined on the space of subsets of F by:

$$\forall K \subset F, \;\; W(K) = \bigcup_{i \in [0,N]} w_i(K)$$

Definition 4: An IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$ is called *hyperbolic* (or *contractive*) if all functions w_i are contractive. The *contraction ratio* of Ω is the minimum of the contraction ratio of the w_i.

Proposition If an IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$ is contractive, there exists a unique set $A \subset F$, called the *attractor* of the IFS Ω, such that $W(A) = A$.

The uniqueness of an attractor for contractive IFS is a result of the Contractive Mapping Fixed Point Theorem for the mapping W acting on $\mathcal{P}(F), d_H$, which is contractive according to the HAUSDORFF distance d_H:
Definition 5: The HAUSDORFF *distance* between two subsets A and B of F is defined by:

$$d_H(A, B) = \max\left(\max_{x \in A}\big(\min_{y \in B} d(x, y)\big), \max_{y \in B}\big(\min_{x \in A} d(x, y)\big)\right)$$

A.2 Computing the attractor

There are two main techniques to compute the attractor of a contractive IFS:

- **Deterministic method:**
  From any kernel S_0, build the sequence $\{S_n\}$ of subsets of F:

  $$S_{n+1} = W(S_n) = \bigcup_{i=1}^{N} w_i(S_n)$$

  For large values of n, S_n is an approximation of the actual attractor of Ω.

- **Stochastic method (toss-coin):**
  Let x_0 be the fixed point of one of the w_i functions. Build the sequence of points 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 Ω. The larger n, the more precise the approximation.

A.3 Inverse problem

The inverse problem for 2D IFS can be stated as follows:

> *Find a contractive IFS, the attractor of which is exactly a given shape (a binary image).*

However, this problem is generally relaxed into:

> *Find a contractive IFS the attractor of which is as close as possible of a given shape for a pre-defined distance function.*

A tool that is usually used for the simplification of this problem is the *Collage theorem*, which states that finding an IFS Ω whose attractor is close to a given shape I, is equivalent to minimising the distance $d_H\big(I, \bigcup_{i=1}^{N} w_i(I)\big)$ with the constraint that all w_i are contractive functions.

Collage theorem [3]
Let A be the attractor of the hyperbolic IFS $\Omega = \{F, (w_i)_{i=1,..,N}\}$, and λ the contraction ratio of Ω. Then:

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

However, some difficulties arise when $d_H\big(I, \bigcup_{i=1}^{N} w_i(I)\big)$ is to be minimised:

- The fitness depends on the contractance of the mappings; if one of the mappings is poorly contractive (i.e. λ close to 1), then the term $\frac{1}{1-\lambda}$ may become very large, thus depriving the bound of much of its sense. In the case of affine IFS, it is possible to estimate λ and thus to minimise $\frac{1}{1-\lambda} d_H\big(I, \bigcup_{i=1}^{N} w_i(I)\big)$ to overcome this difficulty. However, for non-linear IFS, the contraction ratio may not be uniform over the domain F which makes it almost impossible to estimate.

- Computing the HAUSDORFF distance itself is CPU-time consuming. Moreover, the HAUSDORFF distance often is counter-intuitive: Figure 12 presents two pairs 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.

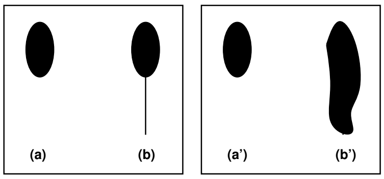


Figure 12: d_H[(a), (b)] = d_H[(a'), (b')] though the shapes look quite different !

These drawbacks led some authors of this paper [16] to consider a fitness function based on the toss-coin algorithm and on more intuitive distances between shapes (namely pixels differences or Euclidian distance), instead of the HAUSDORFF distance.

Section 4 demonstrates how the Parisian approach allows one to use informations stemming from both collage theorem and toss-coin algorithm in order to solve the inverse problem.

B A counter-example : local contractance is not equivalent to global contractance

$$\text{If} \quad x <= -1 \qquad w_0(x,y) = \begin{pmatrix} 0.5x - 1 \\ 0.5y \end{pmatrix}$$

$$\text{If} \quad 1 <= x <= -1 \quad w_0(x,y) = \begin{pmatrix} 0.25(x^2 - 3x - 10) \\ 0.25(3+x)y \end{pmatrix}$$

$$\text{If} \quad x >= 1 \qquad w_0(x,y) = \begin{pmatrix} x - 4 \\ y \end{pmatrix}$$

$$\text{If} \quad x <= -1 \qquad w_1(x,y) = \begin{pmatrix} x + 4 \\ y \end{pmatrix}$$

$$\text{If} \quad 1 <= x <= -1 \quad w_1(x,y) = \begin{pmatrix} 0.25(-x^2 - 3x + 10) \\ 0.25(3-x)y \end{pmatrix}$$

$$\text{If} \quad x >= 1 \qquad w_1(x,y) = \begin{pmatrix} 1 + 0.5x \\ 0.5y \end{pmatrix}$$

Functions w_0 et w_1 are continuous. Their fixed points are respectively $P_0 = (-2,0)$ and $P_1 = (2,0)$, and they satisfy:

$$\forall X \in [-4,4] \times [-4,4], \quad d\big(P_n, w_n(X)\big) \leq sd(P_n, X) \quad \text{with} \quad s < 1$$

They are however not contractive and do not define a unique attractor for the $\{w_0, w_1\}$ IFS: there exists several sets, such that $A = W(A) = w_0(A) \cup w_1(A)$ (see figure 13) and the $W^n(X)$ sequences do not always converge when $n \to \infty$ (see figure 14).

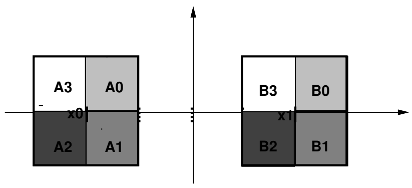


Figure 13: $W(A0 \cup B0) = A0 \cup B0$, $W(A1 \cup B1) = A1 \cup B1$, $W(A2 \cup B2) = A2 \cup B2$, $W(A3 \cup B3) = A3 \cup B3$, and $W(x_0 \cup x_1) = x_0 \cup x_1$.

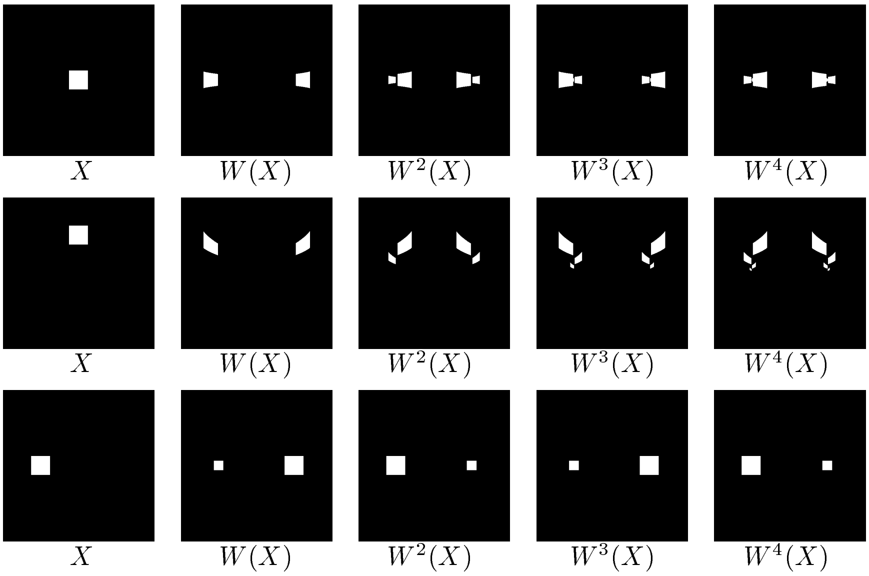


Figure 14: Three examples for which the sequence $W^n(X)$ diverges.