A Comparison of Different Evolutive Niching Strategies for Identifying a set of Selfsimilar Contractions for the IFS Inverse Problem

María L. Ivanissevich
Universidad Nacional de la Patagonia Austral,
Río Gallegos, Argentina

A. S. Cofiño and J.M. Gutiérrez
Dept. of Applied Mathematics, University of Cantabria,
Santander, Spain

gutierjm@unican.es,
http://personales.unican.es/~gutierjm

Abstract

The key problem in fractal image compression is that of obtaining the IFS code (a set of linear transformations) which approximates a given image with a certain prescribed accuracy (inverse IFS problem). In this paper, we analyze and compare the performance of sharing and crowding niching techniques for identifying optimal selfsimilar transformations likely to represent a selfsimilar area within the image. The best results are found using the deterministic crowding method. We also present an interactive Matlab program implementing the algorithms described in the paper.

Keywords

Evolutive algorithms, fractals, iterated function system (IFS).

1 Introduction

In the last two decades Iterated Function Systems (IFS) have been established as intuitive and flexible fractal models in several areas of computer graphics [1]. The main features of IFS models are their simplicity and mathematical soundness: An IFS consists of a set of contractive affine transformations, which express a unique image (the attractor) in terms of selfsimilarities in a simple geometric way. These models have been applied to many interesting problems, including fractal image compression [2]. This possibility is provided by the Collage Theorem [3], which shows that any image can be closely approximated by the attractor of an appropriate IFS model (encoded only with a few parameters, the coefficients of the affine transformations).

The key problem in fractal image compression is that of obtaining the IFS code which approximates a given image with a certain prescribed accuracy (inverse IFS problem). Some attempts to solve this problem include "moment matching", which reduce the problem to solving a system of equations [4]; wavelets transforms, which use similarity preserving properties of continuous wavelets to find the appropriate transformations [5], etc., but none of these techniques have prove to be efficient in general.

Other recent attempts to solve the inverse problem use *evolutive algorithms*, a new optimization paradigm that models a natural evolution mechanism (see [6] for an introduction to this field). Evolutive algorithms work with a population of individuals (in this case a population of IFS models) each one them representing a search point in the space of potential solutions of the inverse problem. The population is able to adapt towards the optimum by means of a random process of selection, and the application of genetic operators, such as recombination and mutation. Several schemes based on this idea have been recently proposed; they only differ in the manner information is encoded and in the specific genetic operators applied in the evolutive process. Two broadly accepted schemes are Genetic Algorithms (GAs) [7, 8] and Evolutionary Strategies (ESs) [9].

Attempts for solving the inverse problem using GAs [10, 11, 12] and ESs [13, 14] have achieved relative success. These algorithms work with a population of IFS models and, therefore, perform a global optimization of the whole set of selfsimilar contractions of the given image. It has been recently proposed [15] a more efficient approach to this problem using an hybrid evolutive-genetic algorithm in two steps:

Step 1: An evolutive strategy is used for identifying selfsimilar contractive transformations of a given image (the problem at this stage is obtaining the selfsimilar structures within the image); the algorithm works with a population of affine transformations.

Step 2: An initial population of IFS models is created by randomly combining the obtained selfsimilar transformations according to their fitness (selfsimilarity degree), and a genetic algorithm is conducted to search the optimal IFS model among the different combinations.

The main idea of the hybrid algorithm is using an appropriate set of transformations (representing selfsimilar areas of the image) to form the IFS models, instead of considering an initial random population. Gutiérrez et al. [15] use a simple $(\mu + \lambda)$-ES algorithm for identifying the set of selfsimilar transformations. However, the evolution of standard ESs is quickly attracted to one of the local maxima of the fitness space forcing to perform successive independent runs to find different selfsimilar transformations. Niching methods [8] help to allow the concurrent existence of different solutions in the same evolutive population.

In this paper, we analyze and compare the performance of sharing and crowding niching techniques applied to this problem. These techniques have been implemented in Matlab program *"Evolutive IFS"*, which allows us comparing and understanding the different niche algorithms in an easy and interactive form.

This paper is organized as follows. In Section 2 we introduce IFS and describe the inverse problem. Some terminology and definitions on ESs are presented in Section 3. Section

2 Iterated Function Systems

An IFS is a set of affine contractive functions t_i, $i = 1, \ldots, n$, which transform a subset of the plane $S \subset \mathbb{R}^2$ onto smaller subsets $t_i(S)$. Then, it is possible to define one transformation, T, in the subsets of the plane by

$$T(S) = \bigcup_{i=1}^{n} t_i(S). \tag{1}$$

For an input set S, we compute $t_i(S)$ for each i, take the union, and get a new set $T(S)$. It can be easily shown that if the t_i are contractive then T is also contractive and has a unique fixed point in the space of all images called the *attractor of the IFS*, $A = T(A)$. Equation (1) gives an intuitive framework for modeling fractal selfsimilar images, since selfsimilarity means that any portion of the object, if enlarged in scale, appears identical to the whole object. This fact is shown in the three examples of Fig. 1, where the boxes indicate each of the selfsimilar portions of the images.

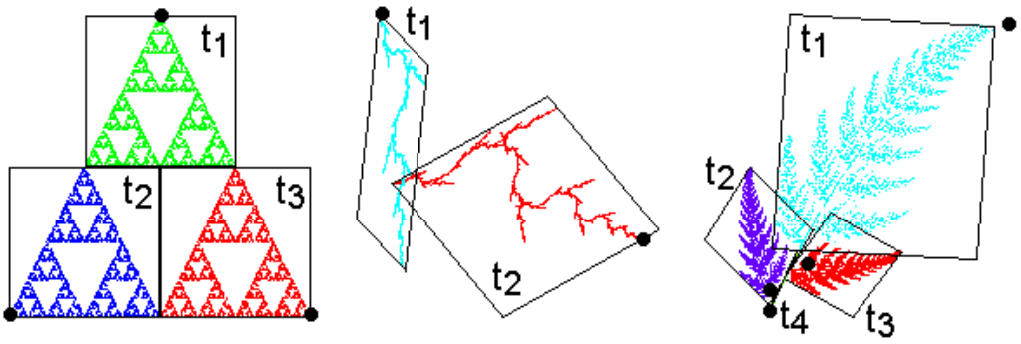

Figure 1: Attractors of three IFS models: The Sierpinsky gasket, a lightning, and a fern. The fixed points of each contraction are also shown.

From this figure we can see that besides mathematical objects, such as the Sierpinsky gasket, selfsimilarity is also present in many real-world patterns; for instance the IFS model resembling the lightning shown Fig. 1 is given by the following transformations:

$$\begin{aligned} t_1(x,y) &= (0.424x - 0.651y + 3.964 \quad , \quad -0.485x - 0.345y + 4.222), \\ t_2(x,y) &= (-0.080x - 0.203y - 4.092 \quad , \quad -0.743x + 0.205y + 3.957). \end{aligned}$$

Each function in an IFS has six degrees of freedom, which can be represented in a number of equivalent forms, such as:

$$t_i(x,y) = \begin{pmatrix} a_{i1} & a_{i2} \\ a_{i3} & a_{i4} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix} \tag{2}$$

$$= \begin{pmatrix} r_{i1}\, cos\theta_{i1} & -r_{i2}\, sin\theta_{i2} \\ r_{i1}\, sin\theta_{i1} & r_{i2}\, cos\theta_{i2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix}. \tag{3}$$

Some of these representations are particularly suitable for the application of crossover and mutation genetic operators. In particular, in this paper we chose the representation given in (3), consisting of parameters $(r_{i1}, r_{i2}, \theta_{i1}, \theta_{i2}, b_{i1}, b_{i2})$ with an intuitive geometric meaning (see [14] for a detailed analysis of several representations).

Generating the attractor of an IFS model is an easy task (see [16]); however, the inverse problem is a hard one. The "Collage Theorem" establishes a condition for a given image I to be approximated by the attractor image A of an IFS model. This theorem gives an upper bound for Haussdorff distance between both images $d(A, I)$ by using I and the transformations t_i forming the IFS:

$$d(A, I) \leq \frac{1}{1-c} d(I, \bigcup t_i(I)), \tag{4}$$

where c is the contraction factor of the IFS formed by $t_1, \ldots, t_n$. This theorem gives a method for solving the IFS inverse problem by means of the following optimization problem:

$$\text{(OPT.1)} \quad \begin{cases} \text{Minimice} \;\; f(\mathbf{t}) = d(I, \cup_{i=1}^{n} t_i(I)), \\ \text{Subject to} \;\; \mathbf{t} = (t_1, \ldots, t_n), \;\; \text{being contractions.} \end{cases} \tag{5}$$

We can also consider the simpler problem of obtaining a single selfsimilar transformation t (as those shown in Fig. 1 for the Sierpinsky, lightning and fern IFS models). In this case we have the following optimization problem:

$$\text{(OPT.2)} \quad \begin{cases} \text{Minimice} \;\; f(t) = d(I, t(I)), \\ \text{Subject to} \;\; t, \;\; \text{being a contraction.} \end{cases} \tag{6}$$

Evolutive algorithms have been quite successful on solving these kind of optimization problems, where standard mathematical algorithms are hard to apply.

3 Evolutive Algorithms and the Inverse IFS Problem

Evolutive algorithms work with a population of individuals which are iteratively adapted towards the optimum by means of a random process of selection, recombination and mutation. During this process, a fitness function measures the quality of the population, and selection favors those individuals of higher quality. Most of the evolutionary algorithms described in the literature for solving the IFS inverse problem follow the optimization problem (OPT.1); in this case, each individual is an IFS model consisting of a number of transformation and its fitness is given by some convenient measure of similarity between the target image and the IFS attractor.

However, as shown by Gutiérrez et al. [15], the inverse problem can be solved more efficiently by first obtaining an appropriate set of transformations by solving (OPT.2) and then using the obtained transformations to feed an initial population in (OPT.1). This method is illustrated in Figure 2, which shows the results of theses two steps when applied to the Sierpinsky carpet (see [15] for details about the implementation).

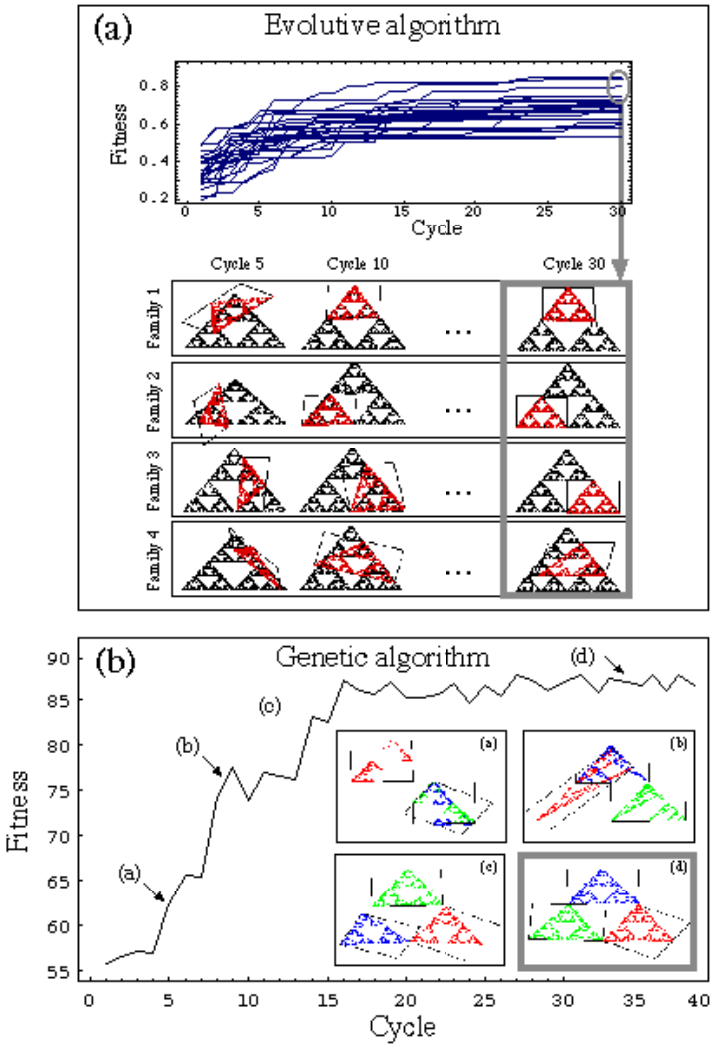

Figure 2: Two-steps hybrid algorithm.

In this paper we shall analyze in detail (OPT.2), improving the simple ES used in [15] by considering different niching schemes. We first describe the particular definition of the components of ESs used in this paper:

- *A convenient coding of individuals.* The population if now formed by linear contractive transformations. For each individual t_i, we consider a vector of real numbers of the form

$$(r_{i1}, r_{i2}, \theta_{i1}, \theta_{i2}, b_{i1}, b_{i2},\ \sigma_{i1}, \sigma_{i2}, \sigma_{i3}, \sigma_{i4}, \sigma_{i5}, \sigma_{i6}).$$

  where the first six components are the parameters of the transformation (individual parameters) and the last six components correspond to standard deviations for individual mutations (strategy parameters).

- *A reproduction mechanism.* Starting from an initial random population obtained by constraining the scaling factors to be lower than one and the translations to be in the range of the figure dimensions, the ES proceeds by iteratively reproducing the population individuals, by the simple criterion of proportionality to their fitness, $P(t_k) = f(t_k)/\sum_i f(t_i)$, together with a linear scaling according to the best and worst individuals. The genetic operators act on the reproduced individuals obtaining a new population.

  In order to deal with the constraints given in (6), each of the transformations is checked to be a contraction and in case it violates this condition, the individual is assigned a negative fitness value, discarding it from the new population.

- *Recombination and Mutation operators.* We use discrete recombination for individual parameters (an offspring individual is formed by selecting at random the components from either the first or second parent) and intermediate recombination for strategy parameters (the standard deviations of an offspring random values between the corresponding components of the parents).

  Mutation is applied to the individual parameters of a transformation t_i, by adding an individual $(0, \sigma_{ik})$ normally distributed random number to the k-th parameters. No mutation is applied to the strategy parameters.

- *A fitness positive function* to be maximized. For a given transformation t_i, the fitness function is computed by evaluating the similarity between the original image, I, and the transformed image, $t_i(I)$. The performance of the evolutive algorithm will depend on the definition of a computationally efficient metric for this problem; since the Hausdorff distance may be inefficient for computational purposes, we have considered the simple Hamming distance instead, obtaining satisfactory results.

  In this paper we assume I to be a selfsimilar image, i.e., the attractor of some IFS model. Then, using a normalized Hamming distance, the fitness function $f(t) = 1 - d(I, t(I))$ is known to have a global maximum at 1, and may have several local maxima.

This fact is illustrated in Figure 3, which shows transformations associated with global and local maxima of the fitness function. Note how the transformations shown in Fig. 3(b) correspond to local minima of the Hamming distance, since no tiny modification of the transformation parameters allow decreasing the Hamming distance between the resulting attractor and the Sierpinsky carpet.

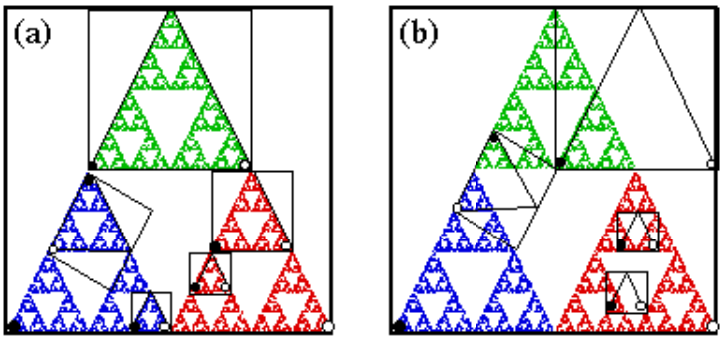

Figure 3: Some transformations of the Sierpinsky gasket corresponding to global (a) and local (b) maximum of the fitness, regarding the Hamming distance used in this paper.

Evolutive strategies allow to find global or local maxima, by using a population with μ parents, with λ offspring for each in every evolutive cycle. (μ, λ) indicates a strategy where parents are discarded from the next generation, whereas strategies of the form $(\mu+\lambda)$ introduces competition between parents and offspring to form a new population (elitism).

Figure 4 shows the results of two independent experiments (families 1 and 2) performing 100 cycles of a (20+10) ES. In the first experiment we get one of the global optima (a selfsimilar region of the image), whereas in the second experiment the algorithm gets stuck into a local maximum.

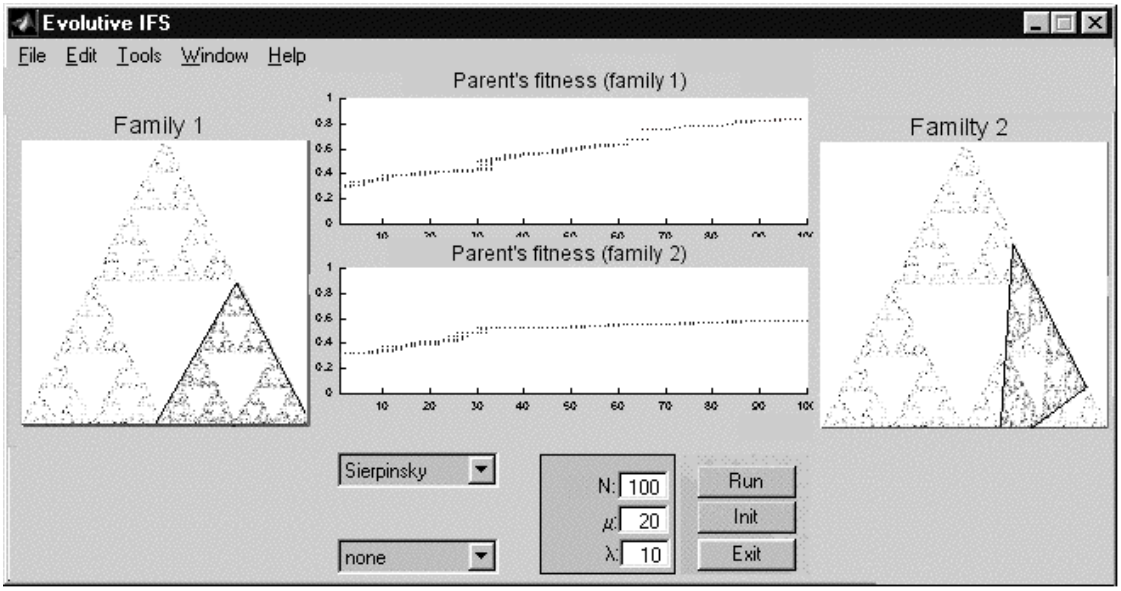

Figure 4: View of the IFS Evolutive Matlab window after running two independent ESs.

We performed several experiments with these ESs and in all cases we found that one of the parents quickly dominates the the population and this finally converges to a single local or global maximum (see Fig. 4). This behavior stimulated the development of algorithms able to form and maintain stable subpopulations (also known as *niches* due to the evolutive metaphor). In the next section, we analyze the performance of the standard niching algorithms when applied to this problem.

4 Niching for Multimodal Fitness Functions

The main goal of niching methods is creating and maintaining several subpopulations, ideally one per local or global maximum of the fitness function, avoiding the convergence of the whole population to a single maximum.

One of the niching methods which has proven effective is *fitness sharing* [8]. Sharing reduces the fitness of the population elements according to the number of individuals concentrated around the given element (a niche), so that the population is balanced among multiple niches. The modified fitness of an individual t_i, called the shared fitness s_f is given by:

$$s_f(t_i) = \frac{f(t_i)}{m(t_i)} \tag{7}$$

where $f()$ is the original fitness function and $m(t_i)$ is a function which determines the niche associated with transformation t_i. In our case, we are interested in spreading the transformations over the original image, so different selfsimilar regions can be found; to this aim we have considered the following sharing function:

$$m(t_i) = \sum_{j=1}^{m} sh(d_{ij}) \tag{8}$$

where d_{ij} is the distance between the translation vectors of transformations t_i and t_j and

$$sh(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{\sigma_s} & \text{if } d_{ij} < \sigma_s \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

A difficulty of this method is choosing an adequate σ_s value, since this requires prior knowledge of the number of maxima and their distribution in the solution space.

Other efficient niching technique is *crowding* [17], which attempts to form and maintain niches by replacing population elements with similar individuals. The probabilistic implementation of this method reproduces and kills a fixed proportion of individuals each generation; each new individual must replace one of the existing elements, preferably the most similar one. However, stochastic replacement errors are not desirable and a deterministic implementation is most convenient in this case. Deterministic crowding works by forming $m/2$ pairs from m population elements every cycle. After performing the genetic

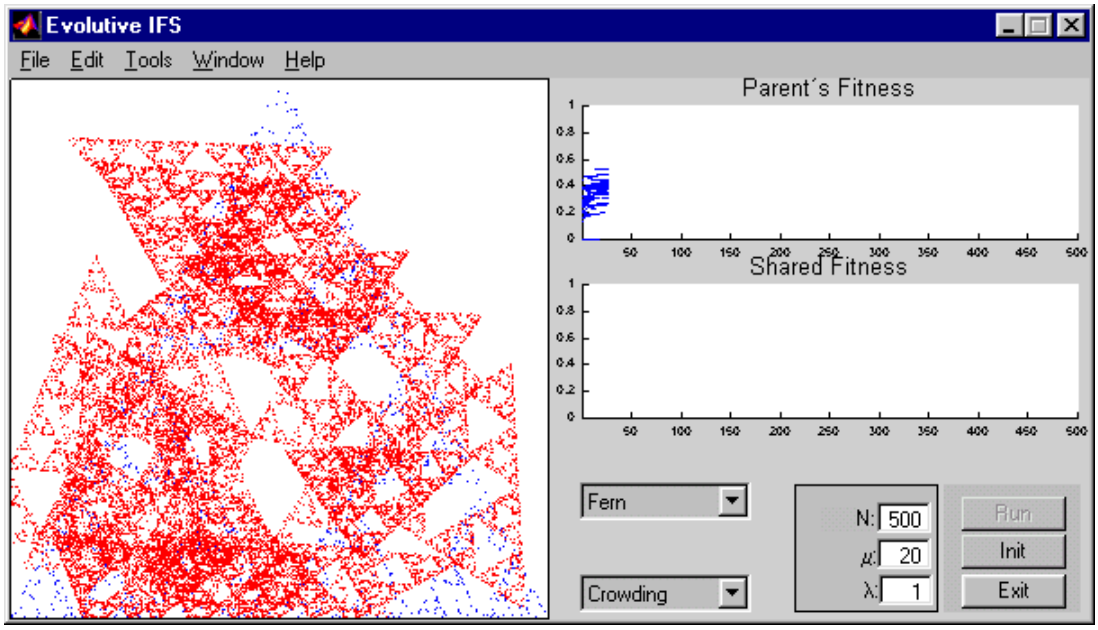

Figure 5: View of the IFS Evolutive Matlab window after 20 cycles of the evolutive process.

operations, each one of the resulting children compete against one of the parents (using a similarity criteria for deciding which one) in order to be included in the population.

In this case, by analogy with the criterion adopted for sharing methods, we have also used the translation vectors of the transformations for calculating similarity.

Several experiments were performed with the aim of comparing the performance of the above niching algorithms. *Sharing* allowed to maintain population diversity, but due to the large number of different local maxima (see Fig. 3) the niches fluctuated on the image and no improvement of the transformations were reached in the long run. The best results were obtained with the *deterministic crowding*, which allowed obtaining in a single run good approximations of the global maxima, as well as other local maxima of the image.

Figure 5 shows a view of the implemented Matlab program after performing 20 cycles of a deterministic crowding algorithm for the Sierpinsky models with a population of 20 individuals. From this figure we can see that, at this stage, the transformations cover almost the window area. Figure 6 shows the results after 500 cycles; it can be seen how the algorithm maintained diversity within the population, allowing to obtain selfsimilar transformations of the model. Therefore, using this algorithm we can obtain in a single run a set of appropriate transformations to feed a genetic algorithm and solve the inverse problem, as described in the introduction of this paper.

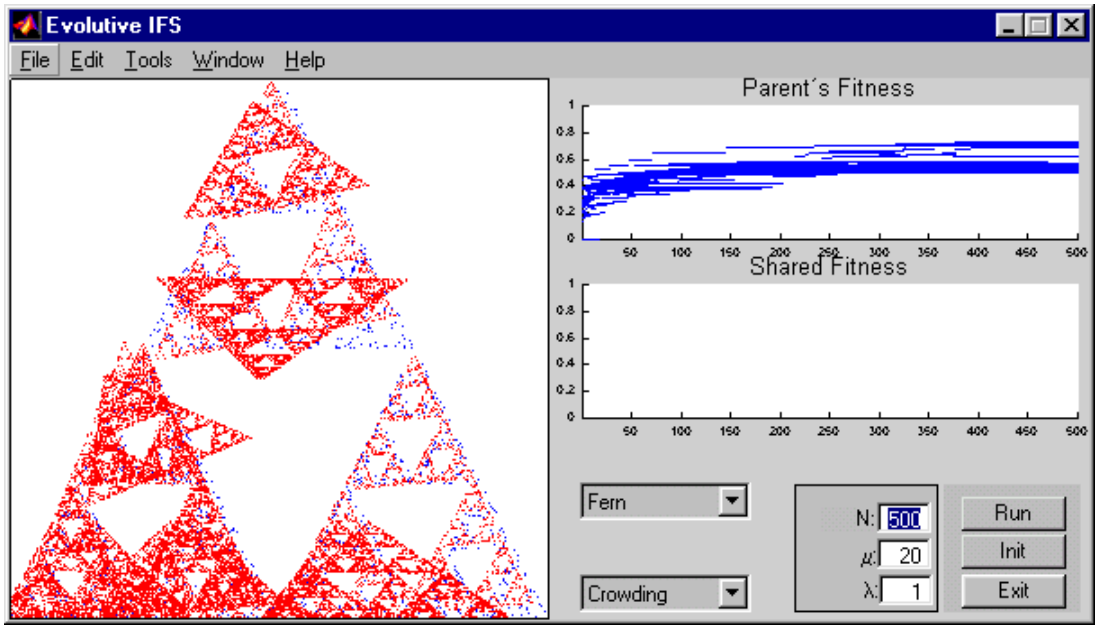

Figure 6: View of the IFS Evolutive Matlab window after 500 cycles of the evolutive process.

References

[1] Turner, M.J. and Blackledge, J.M., Andrews, P.R.: Fractal Geometry in Digital Imaging. Academic Press, 1998.

[2] Fisher, Y.: Fractal Image Compression: Theory and Application. Springer Verlag, 1995.

[3] Barnsley, M.F.: Fractals everywhere, second edition. Academic Press, 1990.

[4] Abiko, T., Kawamata, M.: IFS coding of non-homogeneous fractal images using Gröbner basis. Proc. of the IEEE International Conference on Image Processing (1999) 25–29.

[5] Berkner, K.: A wavelet-based solution to the inverse problem for fractal interpolation functions, in L. Véhel et al. editors. Fractals in Engineering'97. Springer Verlag, 1997.

[6] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, second edition, Springer-Verlag, 1994.

[7] Holland, J.H.: Adaptation in natural and artificial systems. The University o Michigan Press, 1975.

[8] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[9] Rechenberg, I.: Evolution strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog Verlag, 1973.

[10] Lutton, E. et al.: Mixed IFS - resolution of the inverse problem using genetic programming. INRIA Rapport 2631, 1995.

[11] Shonkwiler, R., Mendivil, F., Deliu, A.: Genetic algorithms for the 1-D fractal inverse problem. Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, 495–501, 1991.

[12] Goentzel, B.: Fractal image compression with the genetic algorithm. Complexity International **1**, 111-126, 1994.

[13] Nettleton, D.J., Garigliano, R.: Evolutionary algorithms and a fractal inverse problem. Biosystems **33**, 221-231, 1994.

[14] Evans, A.K. and Turner, M.J.: Specialisation of evolutionary algorithms and data structures for the IFS inverse problem, in M.J. Turner editor. Proceedings of the Second IMA Conference on Image Processing: Mathematical Methods, Algorithms and Applications, 1998.

[15] Gutiérrez, J.M., Cofiño, A.S., and Ivanissevich, M.L. An Hybrid Evolutive–Genetic Strategy for the Inverse Fractal Problem of IFS Models. in Lecture Notes in Artificial Intelligence, Springer-Verlag, in press.

[16] Gutiérrez, J.M., lglesias, A. ,Rodríguez, M.A. and Rodríguez, V.J.: Generating and Rendering Fractal Images. The Mathematica Journal **7(1)**, 6–14, 1997.

[17] Mahfoud, S.W.: Crowding and preselection revisited, in Proc. PPSN-92, Elsevier, 1992.

[18] Yin, X. and Germay, N.: , A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization, Proc. I.C. Artificial Neural Nets and Genetic Algorithms, 450–457, 1993.