Modified OFS-RDS Bat Algorithm for IFS Encoding of Bitmap Fractal Binary Images

Abstract

This work is an extension of a previous paper (presented at the Cyberworlds 2019 conference) introducing a new method for fractal compression of bitmap binary images. That work is now extended and enhanced through three new valuable features: (1) the bat algorithm is replaced by an improved version based on optimal forage strategy (OFS) and random disturbance strategy (RDS); (2) the inclusion of new similarity metrics; and (3) the consideration of a variable number of contractive maps, whose value can change dynamically over the population and over the iterations. The first feature improves the search capability of the method, the second one improves the reconstruction accuracy, and the third one computes the optimal number of contractive maps automatically. This new scheme is applied to a benchmark of two binary fractal images exhibiting a complex and irregular fractal shape. The graphical and numerical results show that the method performs very well, being able to reconstruct the input images with high accuracy. It also computes the optimal number of contractive maps in a fully automatic way. A comparative work with other alternative methods described in the literature is also carried out. It shows that the presented method outperforms the previous approaches significantly.

Keywords: swarm intelligence, bat algorithm, fractal compression, iterated function systems, bitmap images

1. Introduction

1.1. Motivation

Image compression has been a very active field of research for decades. However, the increasing volume of traffic and sharing of online video and image content has led to an impressive resurgence of interest in image compression. Its primary goal is to reduce the cost for storage and/or transmission of digital images by taking advantage of internal redundancies in the images. This technology provides efficient storage of digital images as well as fast and reliable transmission of images among different devices and over the Internet. There are many techniques available for digital image compression [17, 32, 34]. In this paper, we focus on fractal image compression, a lossy compression technique based on the fractal geometry that relies on the fact that, very often, some parts of an image

resemble other parts of the same image. This feature is also characteristic in fractal geometry, as the fractal objects exhibit the property of *self-similarity*: they show (at least, approximately) similar patterns at different scales [12].

The core idea of fractal compression is to identify similar parts within a digital image and then, compute affine transformations connecting them, so that the image can be (approximately) reconstructed through iterative application of such transformations on an initial image [6, 13]. This is done through the so-called iterated functions systems (IFS). Basically, an IFS is a finite set of contractive affine maps $\{\phi_i\}_{i=1,\ldots,\eta}$, defined on a complete metric space, $\mathcal{M}$. These affine maps depend on several parameters accounting for different 2D geometrical operations (scaling, rotation, shearing, and translation). The collection of suitable values of such parameters for all affine maps of the IFS used to reconstruct a given digital image is called the IFS code of the image.

The iterated function systems were developed by Hutchinson in [23]. He proved that any IFS on a complete metric space has a unique non-empty compact fixed set (called the *attractor* of the IFS), whose graphical representation is a fractal image. He also defined an iterative way to obtain the attractor of an IFS through the Hutchinson operator (see Sect. 2 for details).

In a previous paper (presented at the Cyberworlds 2019 conference), the authors proposed a modified bat algorithm coupled with a local search heuristics for fractal image compression of bitmap images [15]. The method performs well, with a similarity percentage of about 68% for the test example in the paper, but was also limited in some ways. For instance, it does not compute the number of contractive maps (which is assumed to be an input of the method). On the other hand, although it outperforms previous methods, its accuracy can still be further improved. The present contribution is aimed at improving that work, as explained in next section.

1.2. Main contributions and structure of this paper

In this work, the bat algorithm-based method introduced in our previous conference paper in [15] is extended and enhanced in several ways. The main contributions of this paper can be summarized as follows:

- The local search in [15] is restricted to the neighborhood of the current best solution, which might be far from the global optimum. This problem is overcome through a new local search procedure based on optimal forage strategy (OFS). This strategy promotes the moves with large benefit during the local search of the method, not only those around the current best.
- In the method in [15], only new solutions with a better fitness (positive moves) can be accepted. This limits the exploratory capacity of the method. To overcome this drawback, a new random disturbance strategy (RDS) is applied, with the effect that

negative moves can also be accepted. This strategy avoids the method to get stuck in a local optimum.

- It is convenient to prioritize the exploration in the early stages of the method, in order to cover the entire search space and identify the most promising search areas, and to proceed later with the intensification of the search in those areas. Accordingly, our method includes a new parameter to switch the behavior between early exploration and late intensification through new evolution equations especially tailored for each particular phase.

- In the method in [15], the number of contractive maps (η) is fixed and assumed to be known. In this paper, the value of η can change dynamically over the population and over the iterations. Furthermore, our new method computes the optimal value of η automatically and accurately.

- As a consequence of the previous changes, the elitism and the mutation operators of the method in [15] are no longer necessary, so in our new method they are removed.

- In this work, several similarity functions (i.e., Hamming, intersection, symmetric difference) and other metrics are considered and analyzed in order to get a better insight about the method and its internal operating principles. This analysis provides valuable information to identify limitations of some metrics as well as to determine the best metrics for this problem.

- As a result of all these improvements, we obtain a Hamming similarity percentage of 86% and 92% for the two examples in the paper. To the best of our knowledge, no previous method has reported such high similarity values, even although they generally use simpler examples than those used in this work.

- A comparative analysis with other alternative methods reported in the literature shows that our method outperforms them by a large margin for the examples in this work.

This paper is organized as follows: Sect. 2 discusses the previous work in the field. The basic concepts and definitions needed to follow the paper are presented in Sect. 3. Then, Sect. 4 describes the collage theorem, the theoretical basis of the digital image compression with IFS. The proposed method is explained in detail in Sect. 5. The computational experiments and the main graphical and numerical results are discussed in Sect. 6. The comparative work of our method with other approaches described in the literature and the computational complexity and CPU times are also reported in that section. The paper closes with the conclusions and some ideas for future work in the field.

2. Previous Work

The concept of fractal encoding of images can be traced back to the seminal work in the 1980s by Michael Barnsley, who obtained several patents for fractal image compression based on his developments on iterated function systems (see [5] for details). The theoretical basis of this work was established a bit earlier by Hutchinson in [23], and then, in [3], where the famous collage theorem was presented. The use of fractal transformations to encode images was introduced in [2]. A popular algorithm for fractal images was published in [4]. This work was enhanced with the first automatic algorithm in [24], based on a new concept called partitioned iterated function systems (PIFS). These methods used exhaustive search strategies and thus, they were computationally expensive, leading to low encoding speed. A lot of work has been done to tackle this issue, using quadtrees, rectangular partitions, and triangular partitions, sometimes in combination with clustering. The list of proposed methods is very large to be included here. The interested reader is referred to the review in [33].

Unfortunately, the fractal image compression problem has revealed to be extremely difficult and, except for some particular cases, no general solution has been reported yet. In general, this problem is strongly affected by the encoding/decoding asymmetry: encoding is extremely computationally expensive, owing to the need to find self-similarities in the image. On the contrary, decoding is astonishingly fast. This fact has made this technology impractical for real-time applications. Many attempts have been done to reduce the huge computational time required for the encoding phase. They include moment matching [1, 14, 37], wavelet transforms [7], and gradient search [38]. However, they are still computational expensive and only work properly for particular cases.

It has been observed that fractal image compression can be formulated as an optimization problem. Therefore, it is a good candidate for metaheuristic techniques, such as those typically found in evolutionary computing and swarm intelligence. Genetic algorithms and genetic programming have been applied in [27, 44] to determine the IFS coding of fractal bitmap images. Work about fractal compression using PIFS in combination with genetic algorithms can be found in [39, 40, 43]. Also, an evolutionary algorithm has been applied in [9] for fractal coding of binary images. Fractal image compression with different variations of the particle swarm optimization can be found in [30, 36]. Other examples of these techniques can be found in are [11, 16, 31, 35].

3. Mathematical Background

In this section, we provide the basic concepts and definitions needed to follow the paper. Further details can be found in [5, 10, 13].

Let (X, d) be a metric space, where X is a set and d a distance defined on X. A *contractive map* ϕ on (X, d) is a function $\phi : X \to X$ for which there is a real number

$0 \leqslant k < 1$ such that:

$$d(\phi(x), \phi(y)) \leqslant k.d(x, y) \qquad \forall x, y \in X$$

An important result is the *Banach fixed-point theorem*, which states that every contractive map on a non-empty complete metric space has a unique fixed point. Moreover, given any $x \in X$, the sequence $x, \phi(x), \phi(\phi(x)), \ldots$ resulting from composing ϕ iteratively with itself, converges to the fixed point.

Let $\mathcal{M} = (\mathbf{\Omega}, \mathbf{\Psi})$ be a complete metric space, where $\mathbf{\Omega} \subset \mathbb{R}^n$, and $\mathbf{\Psi}$ is a distance on $\mathbf{\Omega}$. An *IFS (iterated function system)* is a finite set $\{\phi_i\}_{i=1,\ldots,\eta}$ of contractive affine maps $\phi_i : \mathbf{\Omega} \longrightarrow \mathbf{\Omega}$ defined on $\mathcal{M}$. In this paper, the IFS will be denoted as $\mathcal{W} = \{\mathbf{\Omega}; \phi_1, \ldots, \phi_\eta\}$. Since in this paper we are focused on 2D bitmap images, from now on we consider the complete metric space $(\mathbb{R}^2, d_2)$, where d_2 denotes the Euclidean distance. Therefore, the affine maps ϕ_κ are bivariate functions given by:

$$\begin{bmatrix} \xi_1^* \\ \xi_2^* \end{bmatrix} = \phi_\kappa \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} \theta_{11}^\kappa & \theta_{12}^\kappa \\ \theta_{21}^\kappa & \theta_{22}^\kappa \end{bmatrix} . \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \begin{bmatrix} \sigma_1^\kappa \\ \sigma_2^\kappa \end{bmatrix} \tag{1}$$

This expression can be written in vector notation as: $\mathbf{\Phi}_\kappa(\mathbf{\Xi}) = \mathbf{\Theta}_\kappa.\mathbf{\Xi} + \mathbf{\Sigma}_\kappa$, where $\mathbf{\Theta}_\kappa$ is a 2×2 matrix describing the rotation, scaling, and shearing operations and $\mathbf{\Sigma}_\kappa$ is a 2D vector describing the translations. Since ϕ_κ is contractive, the eigenvalues of $\mathbf{\Theta}_\kappa$, denoted as $\lambda_1^\kappa, \lambda_2^\kappa$, hold: $|\lambda_j^\kappa| < 1$ and also we have $\mu_\kappa = |det(\mathbf{\Theta}_\kappa)| < 1$. Intuitively, this means that the map ϕ_κ shrinks distances between points.

From Eq. (1), we can see that any contractive affine map ϕ_κ is uniquely defined by the set of parameters $\{\theta_{11}^\kappa, \theta_{12}^\kappa, \theta_{21}^\kappa, \theta_{22}^\kappa, \sigma_1^\kappa, \sigma_2^\kappa\}$. Furthermore, any IFS, say $\mathcal{W}$, is fully characterized by the collection of parameters $\{\theta_{ij}^\kappa, \sigma_i^\kappa\}_{i,j=1,2;\kappa=1,\ldots,\eta}$. This set of parameters is called the *IFS code* of $\mathcal{W}$.

Let $\mathcal{C}_\mathcal{S}(\mathbf{\Omega})$ denote the set of all compact (i.e., closed and bounded) subsets of $\mathbf{\Omega}$. Note that the bitmap images are compact subsets of $\mathbb{R}^2$. The *Hausdorff metric* h on $\mathcal{C}_\mathcal{S}(\mathbf{\Omega})$ is defined as:

$$h(\mathcal{R}, \mathcal{S}) = max\left\{d_h(\mathcal{R}, \mathcal{S}), d_h(\mathcal{S}, \mathcal{R})\right\} \tag{2}$$

where: $d_h(\mathcal{R}, \mathcal{S}) = \max\limits_{x \in \mathcal{R}} \min\limits_{y \in \mathcal{S}} d_2(x, y)$.

It has been proved that, since $(\mathbb{R}^2, d_2)$ is a complete metric space, so is $(\mathcal{C}_\mathcal{S}(\mathbf{\Omega}), h)$ [5]. We can define a transformation, $\mathcal{H}$, called the *Hutchinson operator* on $\mathcal{C}_\mathcal{S}(\mathbf{\Omega})$, as:

$$\mathcal{H}(\mathcal{B}) = \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{B}) \qquad \forall \mathcal{B} \in \mathcal{C}_\mathcal{S}(\mathbf{\Omega}) \tag{3}$$

This operator defines the join action of all contractive maps ϕ_κ. Since all the ϕ_κ are contractions in $(\mathbb{R}^2, d_2)$, $\mathcal{H}$ is also a contraction in $(\mathcal{C}_\mathcal{S}(\mathbf{\Omega}), h)$ [23]. Then, according to the

Banach fixed-point theorem, $\mathcal{H}$ has a unique fixed point, $\mathcal{H}(\mathcal{A}) = \mathcal{A}$. Interestingly, the set $\mathcal{A}$ (called the *attractor of the IFS*) is a fractal image.

Given an IFS with η contractive maps $\{\phi_1, \ldots, \phi_\eta\}$, there are several methods for rendering its corresponding attractor [20]. The most popular one is the *probabilistic algorithm*, where each contractive map ϕ_κ is associated with a probability $\omega_\kappa > 0$, such that $\sum_{\kappa=1}^{\eta} \omega_\kappa = 1$. Starting with an compact set $\mathbf{\Xi}_0 \in \mathbf{\Omega}$, and proceeding iteratively, one of the maps of the IFS is randomly chosen at iteration j with probability ω_κ to yield $\mathbf{\Xi}_j = \phi_\kappa(\mathbf{\Xi}_{j-1})$. The process is repeated again for the resulting set, and so on. It can be proved that $\overline{\{\mathbf{\Xi}_j\}}_j = \mathcal{A}$, meaning that this iterative process can be used to render the attractor [5, 19]. The couple $(\mathcal{W}, \mathcal{P})$ comprised of the IFS, $\mathcal{W}$, and the set of probabilities $\mathcal{P} = \{\omega_1, \ldots, \omega_\eta\}$, is called and IFS with probabilities (IFSP). The initial set $\mathbf{\Xi}_0$ can be any compact set. However, since the maps ϕ_κ are contractive, it is advisable to take $\mathbf{\Xi}_0$ as a single point for computational efficiency.

The set of probabilities, $\mathcal{P}$, plays a significant role for the good performance of the rendering process. Several approaches to compute suitable values for the ω_κ can be found in the literature [12, 18]. The most popular method, called *Barnsley's algorithm* (also, *chaos game*), consists of taking a probability value ω_κ related to the area filled by the contractive map ϕ_κ, which is proportional to its contractive factor, $\mu_\kappa = |det(\mathbf{\Theta}_\kappa)| = |\theta_{11}^\kappa.\theta_{22}^\kappa - \theta_{12}^\kappa.\theta_{21}^\kappa|$. The method then selects:

$$\omega_i = \frac{\mu_i}{\sum_{j=1}^{\eta} \mu_j} \quad ; \quad i = 1, \ldots, \eta. \tag{4}$$

This is also the method used in this paper. Other choices are possible as well, even leading to more efficient values [19]. However, this problem is out of the scope of this work and will not be addressed here.

4. Digital Image Compression with IFS: the Collage Theorem

The starting point for digital image compression with IFS is the *collage theorem*, firstly reported in [3]. Given an IFS, $\mathcal{W} = \{\mathbf{\Omega}; \phi_1, \ldots, \phi_\eta\}$, with contractivity factor $0 < \mu < 1$ (given by $\mu = \max_{\kappa=1,\ldots,\eta} \mu_\kappa$, where μ_κ is the contractivity factor of the map ϕ_κ), and $\mathcal{L}$ a non-empty compact subset $\mathcal{L} \in \mathcal{C}_\mathcal{S}(\mathbf{\Omega})$, if

$$H\left(\mathcal{L}, \mathcal{H}(\mathcal{L})\right) = H\left(\mathcal{L}, \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{L})\right) \leqslant \epsilon$$

for some $\epsilon \geqslant 0$, where $H(.,.)$ is the Hausdorff metric, then

$$H(\mathcal{L}, \mathcal{A}) \leqslant \frac{\epsilon}{1-\mu}$$

where $\mathcal{A}$ is the attractor of the IFS. This is equivalent to say that:

$$H(\mathcal{L}, \mathcal{A}) \leqslant \frac{1}{1-\mu} H\left(\mathcal{L}, \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{L})\right).$$

In practical terms, the collage theorem states that given any (not necessarily fractal) digital image $\mathcal{F}$, there exists an IFS, say $\mathcal{W}$, whose attractor has a graphical representation $\mathcal{F}'$ that approximates $\mathcal{F}$ accurately, according to the Hausdorff metric. In other words, any digital image can be graphically approximated through an IFS.

This theorem defines the basis of any fractal image compression method. To reconstruct a digital image $\mathcal{F}$, we need to obtain the collection of parameters of an IFS (i.e. its IFS coding), providing a good approximation of $\mathcal{F}$ by $\mathcal{H}(\mathcal{F})$. However, it is enough to approximate $\mathcal{F}$ by $\mathcal{H}(\mathcal{I})$, where $\mathcal{I}$ is *any* initial image (note that the attractor of $\mathcal{W}$ is independent of the initial image $\mathcal{I}$). Such approximation must be measured according to a predefined similarity function $\mathcal{S}$ computing the graphical distance between $\mathcal{F}$ and $\mathcal{H}(I)$.

The discussion in previous paragraph means that digital image compression with IFS can be formulated as the following optimization problem:

$$\underset{\{\boldsymbol{\Theta}_\kappa, \boldsymbol{\Sigma}_\kappa, \omega_\kappa\}_{\kappa=1,\ldots,\eta}}{optimize} \quad \mathcal{S}(\mathcal{F}, \mathcal{H}(I)) \tag{5}$$

for some similarity function $\mathcal{S}$. This problem is far from being trivial. To begin, the problem is continuous, because all free variables in $\{\boldsymbol{\Theta}_\kappa, \boldsymbol{\Sigma}_\kappa, \omega_\kappa\}_i$ are real-valued. It is also constrained, because the corresponding functions ϕ_κ have to be contractive. Furthermore, the problem is multimodal, as there can be several global or local optima of the similarity function. Finally, the problem can be high-dimensional for complex images, which might require many contractive maps for accurate reconstruction. Obviously, this problem cannot be solved through classical mathematical optimization techniques. Several alternative techniques have been proposed to tackle this issue, as already described in Sect. 2. However, the problem remains open and new (more powerful) methods are still required to address this problem.

5. The Proposed Method

The proposed method is presented in this section. Firstly, a brief overview of the method is described. Then, the different elements of the method are discussed in detail.

5.1. Overview of the method

The input of our method is a fractal image, $\mathcal{F}$. It is assumed that $\mathcal{F}$ is given as a rectangular binary bitmap image of size $M \times N$ (measured in pixels) on the compact domain $\mathbf{\Omega} = [a, b] \times [c, d] \subset \mathbb{R}^2$. Mathematically, the image is represented by a matrix of 0s and 1s and size $M \times N$, where value 1 means that the corresponding pixel belongs to the image, and 0 otherwise. The goal is to compute the IFS $\mathcal{W} = \{\phi_1, \ldots, \phi_\eta\}$ optimizing the expression (5). Note that this task also implies to obtain the optimal number of contractive functions, η (which was assumed to be known in the previous conference paper). To this aim, we consider an initial population of potential candidate solutions (called individuals or bats), as discussed in Sect. 5.2. Then, the method described in Sect. 5.4 is applied to obtain the solution of our optimization problem using the fitness functions described in Sect. 5.3. Next sections describe the different components of the method in detail.

5.2. Representation of individuals and search space

Evolutionary algorithms always need an adequate representation of the individuals of the population either in the phenotype or in the genotype. In this problem, the phenotype corresponds to a realization of a particular potential solution leading to the attractor of the corresponding IFS. In this work we consider the genotype, given by the chromosomes, a sequence of genes encoding the properties of the individuals. In this method, the population at iteration t is a set of $\mathcal{P}$ individuals (called bats), $\{\mathcal{W}_1^t, \mathcal{W}_2^t, \ldots, \mathcal{W}_\mathcal{P}^t\}$, where each $\mathcal{W}_\nu^t$ is a collection of η_ν contractive maps:

$$\mathcal{W}_\nu^t = \left\{\phi_{1,\nu}^t, \phi_{2,\nu}^t, \ldots, \phi_{\eta_\nu,\nu}^t\right\} \qquad (\nu = 1, \ldots, \mathcal{P}) \tag{6}$$

where:

$$\phi_{i,\nu}^t = \left(\theta_{1,1}^{i,\nu,t}, \theta_{1,2}^{i,\nu,t}, \theta_{2,1}^{i,\nu,t}, \theta_{2,2}^{i,\nu,t}, \sigma_1^{i,\nu,t}, \sigma_2^{i,\nu,t}\right) \qquad (i = 1, \ldots, \eta_\nu) \tag{7}$$

subjected to the constraint that every $\phi_{i,\nu}^t$ must be contractive. This is equivalent to say that the following constraints must hold:

$$\begin{cases} \left(\theta_{1,1}^{i,\nu,t}\right)^2 + \left(\theta_{2,1}^{i,\nu,t}\right)^2 < 1 \\ \left(\theta_{1,2}^{i,\nu,t}\right)^2 + \left(\theta_{2,2}^{i,\nu,t}\right)^2 < 1 \\ \left(\theta_{1,1}^{i,\nu,t}\right)^2 + \left(\theta_{1,2}^{i,\nu,t}\right)^2 + \left(\theta_{2,1}^{i,\nu,t}\right)^2 + \left(\theta_{2,2}^{i,\nu,t}\right)^2 < 1 + \left(\theta_{1,1}^{i,\nu,t}.\theta_{2,2}^{i,\nu,t} - \theta_{1,2}^{i,\nu,t}.\theta_{2,1}^{i,\nu,t}\right)^2 \end{cases} \tag{8}$$

Note that each bat $\mathcal{W}_\nu^t$ may have a different length, η_ν. This is in clear contrast with the work in [15] where all bats are assumed to have the same length. This new feature provides a mechanism to compute the optimal value of η.

These bats $\mathcal{W}_\nu^t$ are initialized with uniform random variables within the search space, given by the compact domain $\mathbf{\Omega} = [a, b] \times [c, d]$. Applying Eq. (1) to the four corner points of the unit square leads to a set of constraints for the free variables. In particular:

$$
\begin{aligned}
&\text{for} \begin{bmatrix} \xi_1^* \\ \xi_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} a \leqslant \sigma_1^{i,\nu,t} \leqslant b \\ c \leqslant \sigma_2^{i,\nu,t} \leqslant d \end{cases} \\
&\text{for} \begin{bmatrix} \xi_1^* \\ \xi_2^* \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} a - \sigma_1^{i,\nu,t} \leqslant \theta_{1,1}^{i,\nu,t} \leqslant b - \sigma_1^{i,\nu,t} \\ c - \sigma_1^{i,\nu,t} \leqslant \theta_{2,1}^{i,\nu,t} \leqslant d - \sigma_1^{i,\nu,t} \end{cases} \\
&\text{for} \begin{bmatrix} \xi_1^* \\ \xi_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \begin{cases} a - \sigma_2^{i,\nu,t} \leqslant \theta_{1,2}^{i,\nu,t} \leqslant b - \sigma_2^{i,\nu,t} \\ c - \sigma_2^{i,\nu,t} \leqslant \theta_{2,2}^{i,\nu,t} \leqslant d - \sigma_2^{i,\nu,t} \end{cases} \\
&\text{for} \begin{bmatrix} \xi_1^* \\ \xi_2^* \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \begin{cases} a \leqslant \theta_{1,1}^{i,\nu,t} + \theta_{2,1}^{i,\nu,t} + \sigma_1^{i,\nu,t} \leqslant b \\ c \leqslant \theta_{1,2}^{i,\nu,t} + \theta_{2,2}^{i,\nu,t} + \sigma_2^{i,\nu,t} \leqslant d \end{cases}
\end{aligned} \tag{9}
$$

In conclusion, the individuals in our population must fulfill the constraints given by Eqs. (8)-(9). These conditions must be checked at every iteration step t.

5.3. The fitness function

Eq (5) requires a similarity function, $\mathcal{S}$, measuring the distance between the attractor of the IFS, given by $\mathcal{H}(I)$, and the original image $\mathcal{F}$. A natural choice is given by the Hausdorff distance between both sets, given by Eq. (2). However, this metric is computational expensive. Furthermore, it is not fully reliable for our goals, as it may identify as similar, images that are actually different geometrically. For these reasons, other similarity functions have been proposed in the literature [9, 15]. In this work, we consider three of them, discussed in following paragraphs.

A classical choice is given by the Hamming distance, Δ. Given two binary images, $\mathcal{F}_1$ and $\mathcal{F}_2$, of the same size $M \times N$ and domain $\mathcal{D}$, they can be represented as two binary matrices of 0s and 1s for the two channel colors. Then, the Hamming distance is given by:

$$
\Delta(\mathcal{F}_1, \mathcal{F}_2) = \sum_{(x,y)\in\mathcal{D}} |\mathcal{F}_1(x,y) - \mathcal{F}_2(x,y)|
$$

where $\mathcal{F}_j(x,y)$ indicates the value (either 0 or 1) of the pixel (x,y) for the image $\mathcal{F}_j$, $j = 1, 2$. From this expression, we can define the *Hamming similarity function*, $\mathcal{S}_\Delta$, as:

$$
\mathcal{S}_\Delta(\mathcal{F}_1, \mathcal{F}_2) = 1 - \frac{\Delta(\mathcal{F}_1, \mathcal{F}_2)}{M \times N} \tag{10}
$$

Note that $\Delta(\mathcal{F}_1, \mathcal{F}_2)$ computes the number of mismatches between both images, and hence, $\mathcal{S}_\Delta(\mathcal{F}_1, \mathcal{F}_2)$ returns the rate of matches relative to the image size. As a result, values of $\mathcal{S}_\Delta(\mathcal{F}_1, \mathcal{F}_2)$ close to 1 mean that the images are very similar, with the value 1 indicating that they are identical.

Other possibility is given by the *intersection similarity function*, $\mathcal{S}_\cap$, given by:

$$\mathcal{S}_\cap(\mathcal{F}_1, \mathcal{F}_2) = \frac{\langle \mathcal{F}_1 \bigcap \mathcal{F}_2 \rangle_\blacksquare}{\langle \mathcal{F}_1 \bigcup \mathcal{F}_2 \rangle_\blacksquare} \tag{11}$$

where $\langle . \rangle_\blacksquare$ represents the number of active (black) pixels of the image.

We can also consider a similarity function based on the symmetric difference between sets, $\ominus$, given by:

$$\mathcal{F}_1 \ominus \mathcal{F}_2 = (\mathcal{F}_1 - \mathcal{F}_2) \bigcup (\mathcal{F}_2 - \mathcal{F}_1)$$

Then, we define the *symmetric difference similarity function*, $\mathcal{S}_\ominus$, as:

$$\mathcal{S}_\ominus(\mathcal{F}_1, \mathcal{F}_2) = \frac{\langle \mathcal{F}_1 \ominus \mathcal{F}_2 \rangle_\blacksquare}{\langle \mathcal{F}_1 \bigcup \mathcal{F}_2 \rangle_\blacksquare} \tag{12}$$

Note that the similarity functions $\mathcal{S}_\Delta$ and $\mathcal{S}_\cap$ lead to maximization problems (the higher, the better), while $\mathcal{S}_\ominus$ corresponds to a minimization problem. However, we remark that, for any two sets A and B, we have: $A \ominus B = (A \bigcup B) - (A \bigcap B)$.Therefore, $\mathcal{S}_\cap$ and $\mathcal{S}_\ominus$ yield complementary results, meaning that one of them can be safely omitted. Therefore, we will report only the results for $\mathcal{S}_\Delta$ and $\mathcal{S}_\cap$, meaning that our problem in Eq. (5) is always a maximization problem on the interval $[0, 1]$.

5.4. Our Approach: Modified OFS-RDS Bat Algorithm

Our previous conference paper in [15] addressed the fractal image compression problem through the bat algorithm, a popular swarm intelligence method for optimization. To this aim, the original bat algorithm was enhanced with three additional features: a new population model with strong elitism, so that the best solutions are preserved to the next generation; new random individuals and mutation operators, to improve the exploratory capacity of the swarm; and a local search heuristics, to strengthen the exploitation phase in the neighborhood of the local optima at later stages. The results in that paper show that the method works well but it can still be further improved in several ways. A critical issue is the optimal number of contractive maps, which is not computed but assumed to be known. This prevents the method to be used in real-world applications, for which this *a priori* knowledge is almost never available. On the other hand, the similarity error can arguably be enhanced through further improvement of the local and global search phases of the bat algorithm.

An interesting variation of the bat algorithm to tackle these issues has been recently proposed in the literature [8]. In that modification, the original bat algorithm is combined with two different strategies: firstly, an optimal forage strategy (OFS) is used to drive the search direction for each bat; then, a random disturbance strategy (RDS) is applied to enhance the global search pattern of the method. These new features are advantageously

used in this work, as explained below.

5.4.1. Original Bat Algorithm

In the original bat algorithm, there are three evolution equations for the position, $\mathbf{x}_i^g$; velocity $\mathbf{v}_i^g$; and frequency, f_i^g, of the i-th individual (bat) at generation g, as follows:

$$f_i^g = f_{min}^g + \beta(f_{max}^g - f_{min}^g) \quad (13)$$

$$\mathbf{v}_i^g = \mathbf{v}_i^{g-1} + [\mathbf{x}_i^{g-1} - \mathbf{x}_*^g]\, f_i^g \quad (14)$$

$$\mathbf{x}_i^g = \mathbf{x}_i^{g-1} + \mathbf{v}_i^g \quad (15)$$

where β is a random variable following the uniform distribution on $[0, 1]$, and $\mathbf{x}_*^g$ denotes the global best position (solution) according to a given fitness function. These equations are updated iteratively in order to look for better solutions in the search space. The position vector is used to encode the potential solutions of the optimization problem under analysis, while the velocity and frequency provide underlying mechanisms to modify the position during the iterative process. To this goal, the bat algorithm considers two search patterns for the bats: a global search, driven with probability r_i^g (called the pulse rate), and a local search with probability $1-r_i^g$. The pulse rate is not constant, but changes over the time according to the equation: $r_i^{g+1} = r_i^0(1 - e^{-\gamma g})$. The global search is modulated through Eqs. (13)-(15), while the local search is driven by a local random walk of the form:

$$\mathbf{x}_i^{g+1} = \mathbf{x}_*^g + \epsilon \mathcal{A}^g \quad (16)$$

with ϵ a uniform random variable on $[-1, 1]$ and $\mathcal{A}^g =< \mathcal{A}_i^g >$ being the average loudness of all the bats at generation g.

Whenever a new solution is better than the previous best one, it is accepted according to a probability that depends on the value of the loudness. If the solution is accepted, the loudness decreases, while the rate of pulse emission decreases. The evolution rule for loudness is: $\mathcal{A}_i^{g+1} = \alpha \mathcal{A}_i^g$ where α is a constant. Typically, each bat has different values for its loudness and pulse emission rate, which are obtained by randomization by taking an initial loudness $\mathcal{A}_i^0 \in (0, 2)$.

5.4.2. OFS-RDS Bat Algorithm

The paper in [8] proposes a modification of the original bat algorithm based on two observations. The first one is that the local search at generation g in Eq. (16) is restricted to the neighborhood of the best solution of the whole swarm, $\mathbf{x}_*^g$. If the global optimum is far from this current best, the local search at that generation becomes essentially useless. This problem can be overcome through a new local search procedure inspired by the optimal forage strategy (OFS). This strategy is driven by a new term called the benefit

of the i-th bat, b_i^g, given by:

$$b_i^g = \frac{f(\mathbf{x}_i^g) - f(\mathbf{x}_i^{g-1})}{||\mathbf{x}_i^g - \mathbf{x}_i^{g-1}||} \tag{17}$$

This benefit term is computed as the ratio between the energy obtained when the bat moves from old position $\mathbf{x}_i^{g-1}$ to new position $\mathbf{x}_i^g$ at iteration g, given by $f(\mathbf{x}_i^g) - f(\mathbf{x}_i^{g-1})$, and the energy invested in the local search, which depends on the distance between both positions, $||\mathbf{x}_i^g - \mathbf{x}_i^{g-1}||$. With this strategy, the method promotes the moves with large benefits during the local search, not merely those based on the current best of the swarm.

The second observation is that any new solution is probabilistically accepted if and only if it is better than the current solution. This limits the exploratory capacity of the swarm, as only positive moves are allowed at the short term, while negative ones are forbidden, even if they become positive at the long term. To overcome this limitation, a random disturbance strategy (RDS) is applied, in which Eq. (14) is replaced by:

$$\mathbf{v}_i^g = \varrho(\mathbf{x}_j^{g-1} - \mathbf{x}_k^{g-1}) \tag{18}$$

where indices j and k are randomly selected from the population, and ϱ is a uniform random variable on the interval $(0, 1)$. The effect of this new equation is to prevent the bat to move within the line between $\mathbf{x}_i^{g-1}$ and $\mathbf{x}_*^g$, forcing it to explore other areas of the search space.

These modifications lead to a new method called OFS-RDS bat algorithm. It has proved to outperform the original bat algorithm on a benchmark of 28 functions proposed as the test suite in the IEEE CEC-2013 competition for real-parameter optimization [8].

5.4.3. Modified OFS-RDS Bat Algorithm

In order to apply the OFS-RDS bat algorithm to the optimization problem of this work, some additional modifications with respect to our previous method in [15] are needed, as follows:

- the elitism of the previous conference paper [15] is advantageous when a fixed number of contractive maps is used. However, in this new approach, this number changes dynamically over the generations, meaning that the global optimum can change drastically, especially at early stages of the iterative process. In this scenario, this elitist procedure becomes useless and, therefore, it has been removed.

- the population model in the previous conference paper is no longer required; the exploratory capacities that it offered are now assumed by the optimal forage and random disturbance strategies.

- the mutation operators in [15] are now replaced by a switching procedure between Eqs. (14) and (18) ressembling that in [8]. At early stages of the bat algorithm,

say T generations, Eq. (18) is preferred in order to explore the search space more efficiently, switching to Eq. (14) for exploitation of the best solution at later stages of the algorithm.

The resulting method is coupled with a local search heuristics for further search intensification in the neighborhood of the global optimum. Similar to [15], in this work we apply the Luus-Jaakola local search procedure [28], as it shows a satisfactory performance. A detailed explanation can be found in [15] and is omitted here to avoid unnecessary duplication of material.

6. Computational Experiments and Results

6.1. Benchmark and Computational Procedure

The method described in the previous section has been applied to two illustrative examples of fractal images. The images, called `blocks` and `bush`, are displayed in Figure 1(top) left and right, respectively. The first one was already presented in the conference paper and is used here for comparative purposes. Both images are generated through the chaos game algorithm with an IFS comprised of five contractive maps, identified with colors blue, dark yellow, red, dark pink, and light green for the `blocks` image, and peach, beige, green, dark magenta, and mustard, for the `bush` fractal. We remark however, that the color is used for better visualization of the contractive maps but does not play any role in the method, which is applied to binary (black and white) images exclusively. The images have been generated with one million points and processed as bitmap images of size 450×450 pixels.

For each example, the input of our method is the bitmap file of the given fractal image, denoted as $\mathcal{I}$ onwards. It consists of a collection of 202,500 pixels, encoded as a numerical binary square matrix of order 450. Numerically, its elements are 0s and 1s, corresponding to the pixels of the fractal image and the pixels of the background, respectively. They are shown in Fig. 1(bottom), with the 1s and 0s represented as black and white pixels, respectively. The number of black and white pixels for the initial input image $\mathcal{I}$, denoted as $\blacksquare_{\mathcal{I}}$ and $\square_{\mathcal{I}}$ respectively, is $\blacksquare_{\mathcal{I}} = 21,790$ and $\square_{\mathcal{I}} = 180,710$.

We apply the method described in Sect. 5 using a population of 100 bats, following the representation described in Sect. 5.2. Each bat corresponds to an IFS comprised of η contractive functions. Different to the work in [15], the value is η is not fixed but allowed to take integer values between 3 and 12. All bats are initialized with random uniform variables for the initial population. Without loss of generality, we can consider the search space to be $\mathbf{\Omega} = [-2, 2] \times [-2, 2]$, i.e., $a = c = -2$, and $b = d = 2$. Each contractive function is subjected to the constraints given by Eqs. (8)-(9). As a result, 100 random initial images are generated for different random numbers of contractive functions. Then,

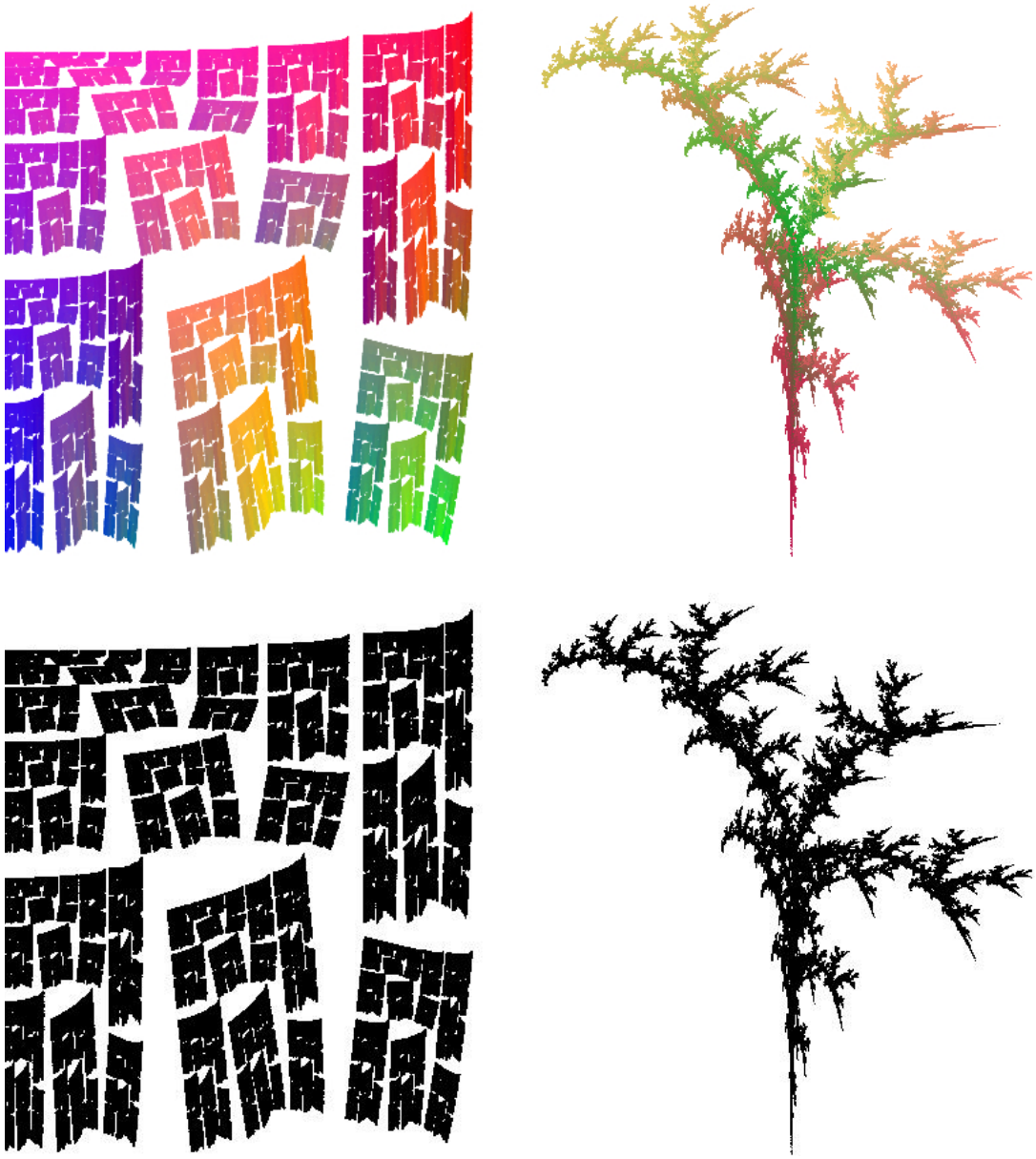

Figure 1: Example images of this paper in colored (top) and binary (bottom) versions: (left) blocks; (right) bush.

the method in Sect. 5.4 is applied for a given number of generations, set to $\mathcal{G}_{max} = 10,000$ in this work, as this value ensures convergence for all examples we tried so far.

Table 1 reports the different parameters of the method and the values used in this paper. The parameter tuning has been discussed in the conference paper and is omitted

Table 1: Modified OFS-RDS bat algorithm parameters and values used in this work.

Symbol	*Meaning*	*Used Value*
$\mathcal{P}$	population size	100
$\mathcal{G}_{max}$	max. number of iterations	10,000
$\mathcal{A}^0$	initial loudness	0.5
$\mathcal{A}_{min}$	minimum loudness	0
r^0	initial pulse rate	0.2
f_{max}	maximum frequency	1.5
α	multiplicative factor	0.3
γ	exponential factor	0.2
T	switching parameter	8,500

here to avoid duplication of material. Still, there is a new parameter, T, called *switching parameter*, to switch between Eqs. (14) and (18), as described in Sect. 5.4.3. It is set to $T = 8,500$ generations in this paper.

6.2. Results

This section discusses the main results of this paper. Firstly, the graphical results are shown. Then, the numerical results are reported and analyzed. Finally, a comparative work with other alternative techniques is discussed.

6.2.1. Graphical results

Since the conference paper showed graphical results for the `blocks` example, the graphical results for the `bush` example are used for our discussion here. Figures 2-5 show the evolution of the best solution of the population for the 10,000 iterations with step-size 250, starting with a random initial image for that example. The images are organized in two parts: on the left, the attractor of the IFS is depicted, with a different color for each contractive map; on the right, this colored attractor is combined with the target image (in black) superimposed on the attractor for better visualization of the difference between both images. These images are also shown in two *QuickTime* videos: *Video1.mov* (length: 84 seconds; size: 1.4 MB) and *Video2.mov* (length: 82 seconds; size: 1.8 MB), submitted as accompanying material of this paper.

The first image in Fig. 2 corresponds to one of the 100 random images in the initial population for the method in this paper. As the reader can see, this initial random image is very far from the target image. As shown in the subsequent images of the sequence in Figs. 2-5, the application of our method reduces this high discrepancy over the iterations, until reaching a very good approximation of the target image. Note also that the number of contractive maps, η, indicated by different colors in the figure, change dynamically

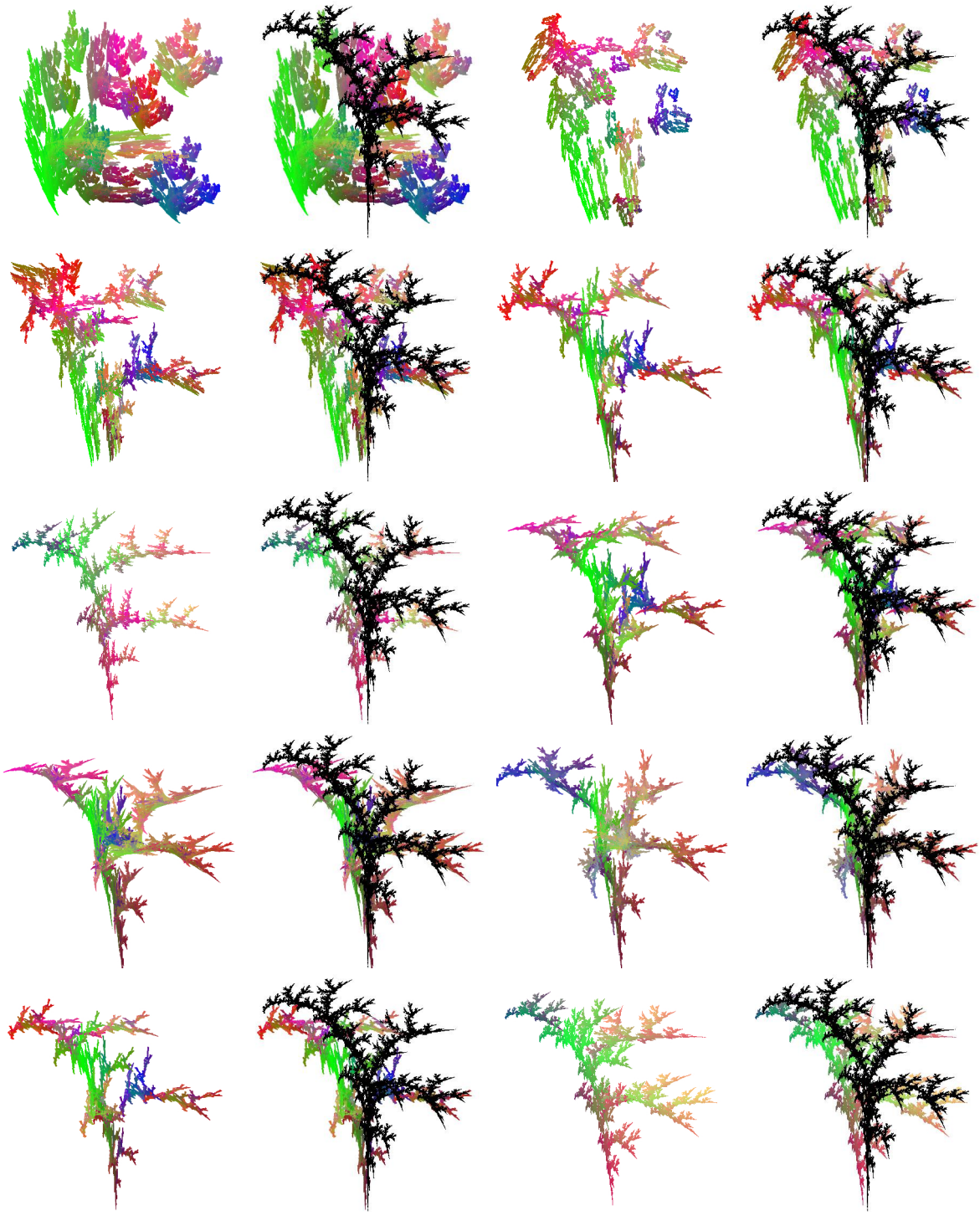

Figure 2: (l-r, t-b) Evolution of the best solution for 0 to 2,250 iterations (step-size 250).

Figure 3: (l-r, t-b) Evolution of the best solution for 2,500 to 4,750 iterations (step-size 250).

Figure 4: (l-r, t-b) Evolution of the population best for 5,000 to 7,250 iterations (step-size 250).

Figure 5: (l-r, t-b) Evolution of the population best for 7,500 to 9,750 iterations (step-size 250).

over the iterations. For instance, the initial random image in Fig. 2 (top–left) has $\eta = 7$ contractive maps, but this value is increased to $\eta = 8$ at iteration $g = 500$ (Fig. 2, second row–left), and decreased to $\eta = 6$ at iteration $g = 1000$ (Fig. 2, third row–left), where the contractive maps in red and blue from the previous image have been automatically removed. Whenever a new contractive map is added, a new color is assigned to the map for better identification. That is the reason why we can see different variations of the color palette throughout Figures 2 to 5.

From Figs. 2–5 we can see that the method is able to approximate the input image with increasing fidelity over the generations. This improvement is visually noticeable by simple observation of the general shape of the global best solution. At initial stages of the iterative process, the shape of the global best changes dramatically, which corresponds to a higher explorative phase, when the method explores the overall search space looking for promising solutions. This variation decreases over the iterations, leading to a more exploitative phase at later stages of the method, when the global shape is slightly modified through small incremental changes in order to enhance local features of the image. As a result, the global best is getting visually closer to the target image, until reaching convergence, when the final attractor image does not change and, consequently, the fitness function value no longer improves.

Figure 6 summarizes the graphical results of the global best of our method for the `blocks` (left) and the `bush` (right) examples in this paper. The top row shows the reconstructed images of the input images in Fig. 1. A simple visual comparison of the original and the reconstructed images shows that the method performs very well, as the final reconstructed images are very similar visually to the input images. The second and third rows of the figure show respectively the union and the intersection sets (displayed in inverted colors for better visualization) of the input and the reconstructed images. These images show that the method is able to capture successfully the major features of the input images even although they exhibit a complicated and irregular fractal pattern. These union and intersection sets will be used in next section to compute the intersection similarity function, $\mathcal{S}_\cap$, and other additional similarity metrics.

Figs. 7–10 (left to right, top to bottom) show the evolution of the union (left) and the intersection (right) sets of the input and the approximating fractal images of the `bush` example for the 10,000 iterations with step-size 250. Note the huge difference between the union and the intersection sets at the early stages of the method. This difference is visually decreasing over the iterations, until the global shape of the input fractal image becomes apparent at later stages of the procedure. Two additional videos, showing respectively the evolution of the union and the intersection sets for the `bush` example, are also submitted as accompanying material of the paper.

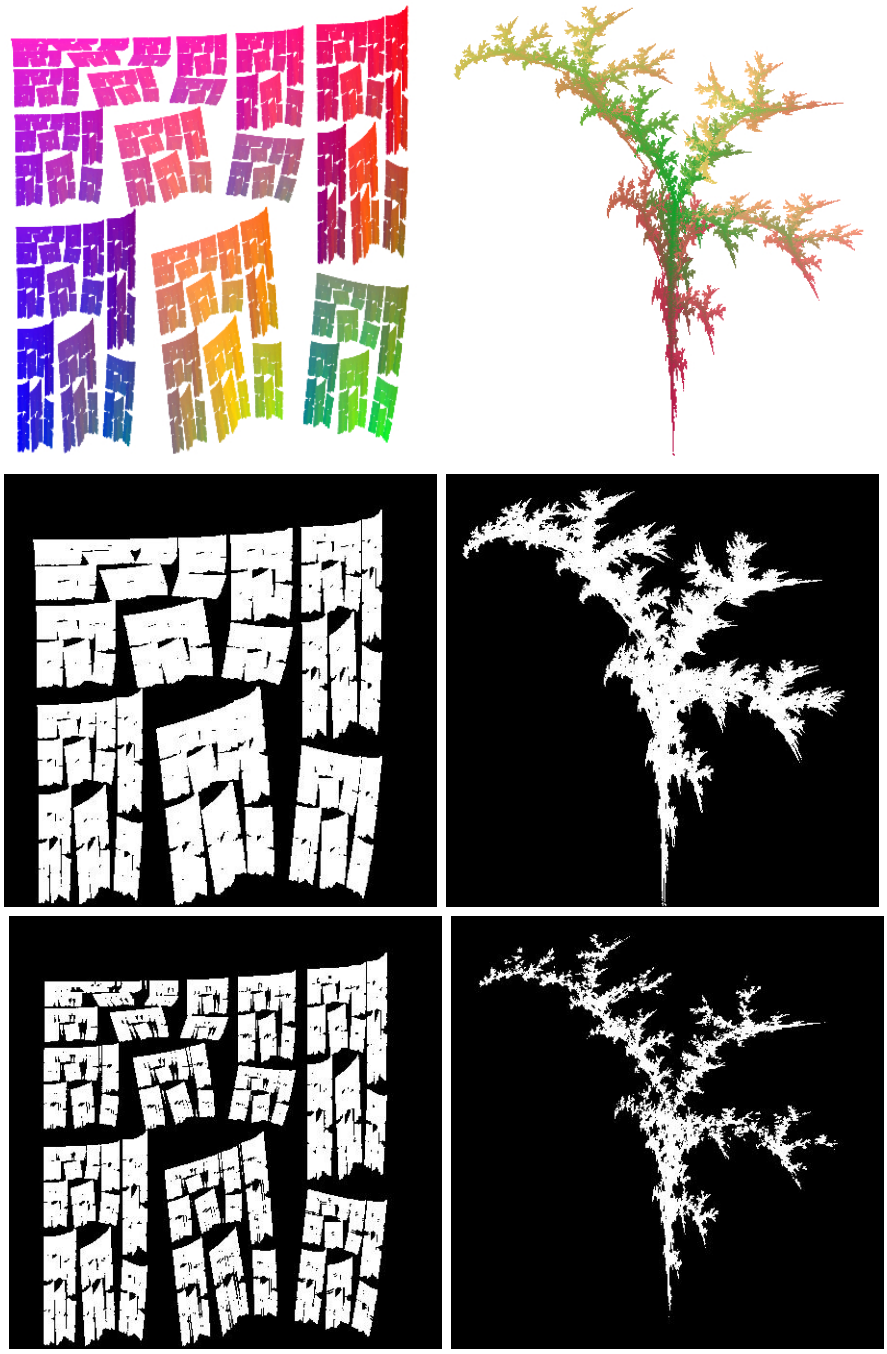

Figure 6: Graphical results of our method for the `blocks` (left) and the `bush` (right) examples in Fig. 1: (top) best reconstructed images; (middle) union and (bottom) intersection sets of the input and reconstructed images in inverted binary colors.

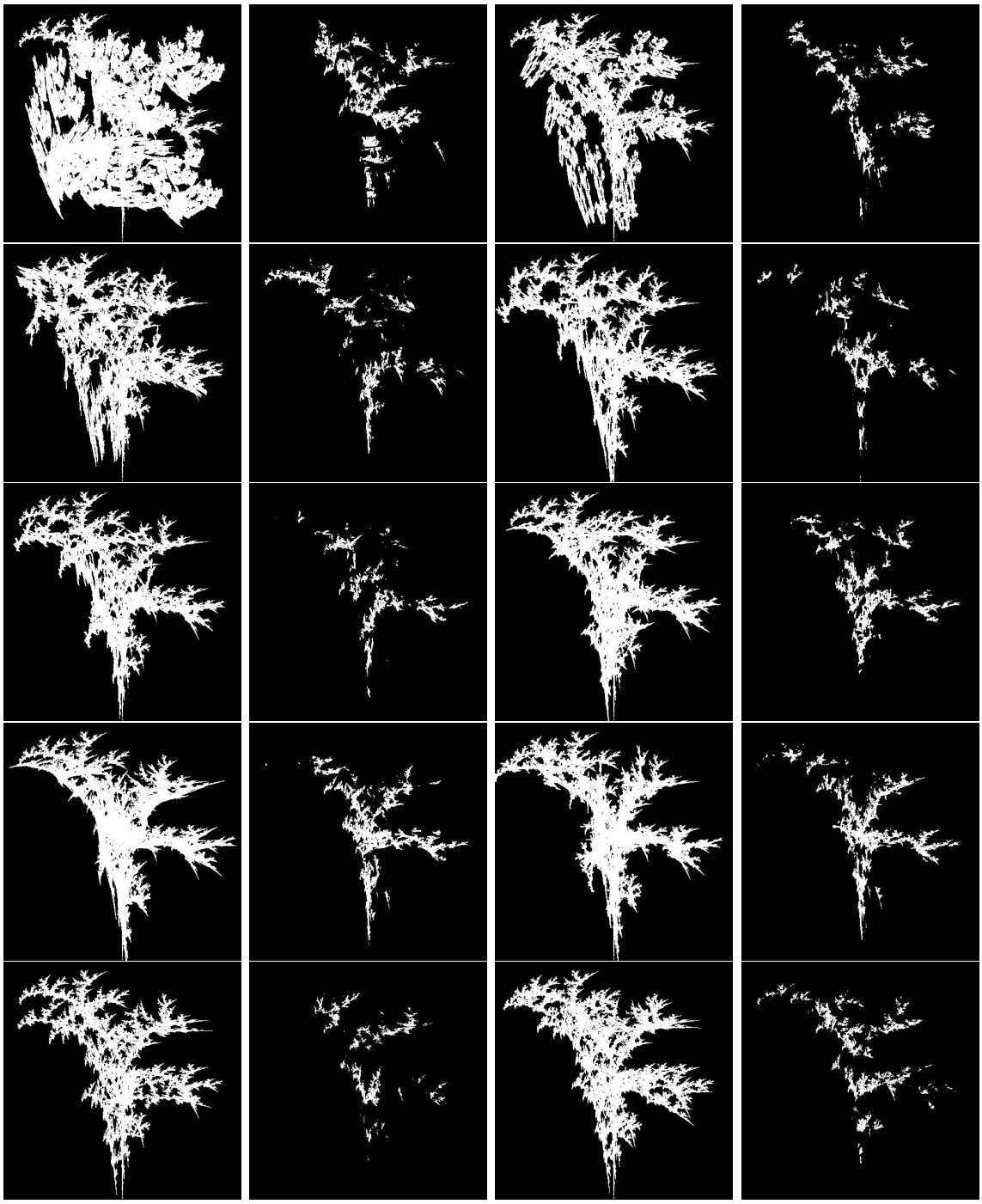

Figure 7: (l-r, t-b) Union (left) and intersection (right) sets of the input and approximating fractal images of the `bush` example for 0 to 2,250 iterations (step-size 250).

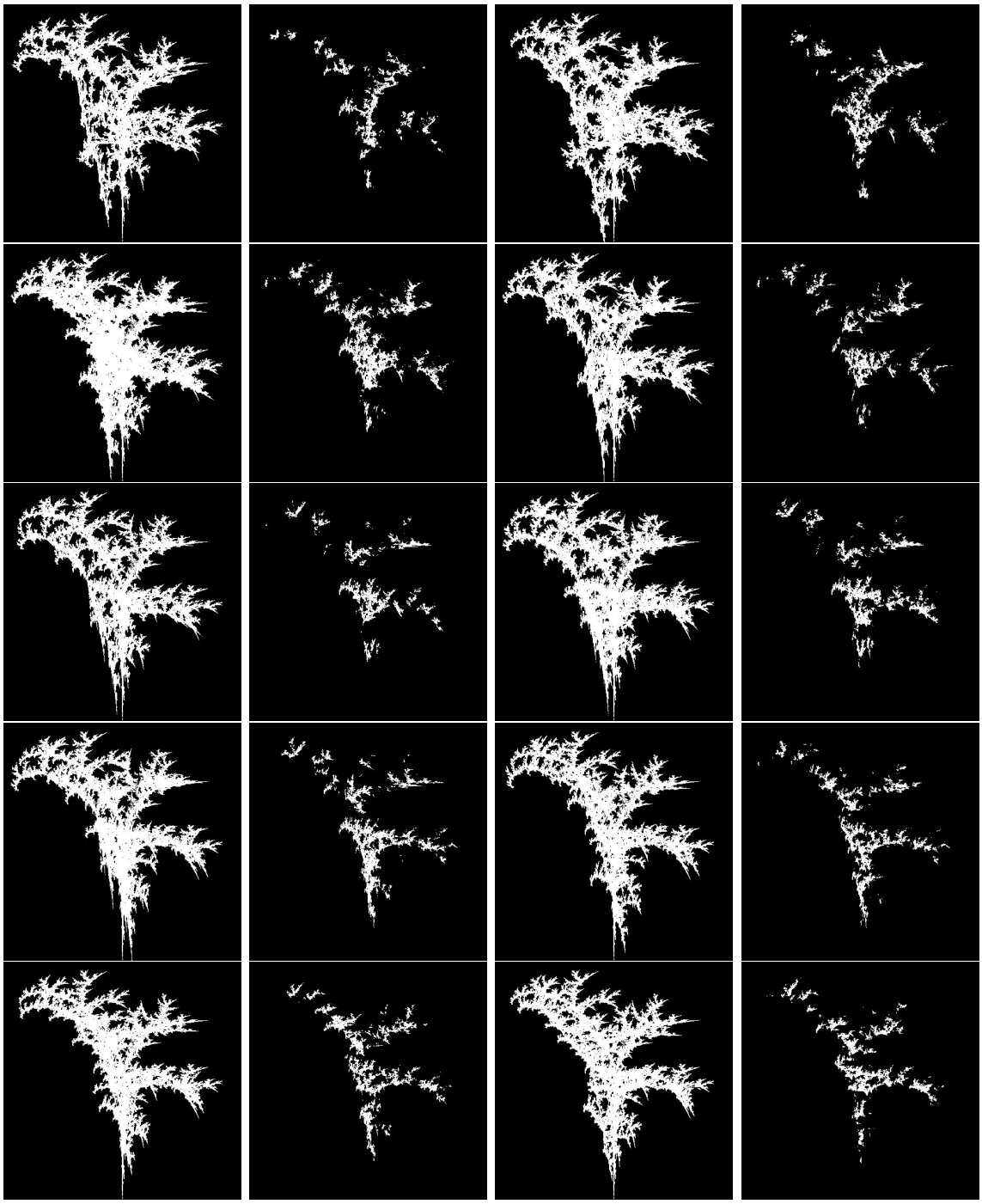

Figure 8: (l-r, t-b) Union (left) and intersection (right) sets of the input and approximating fractal images of the `bush` example for 2,500 to 4,750 iterations (step-size 250).

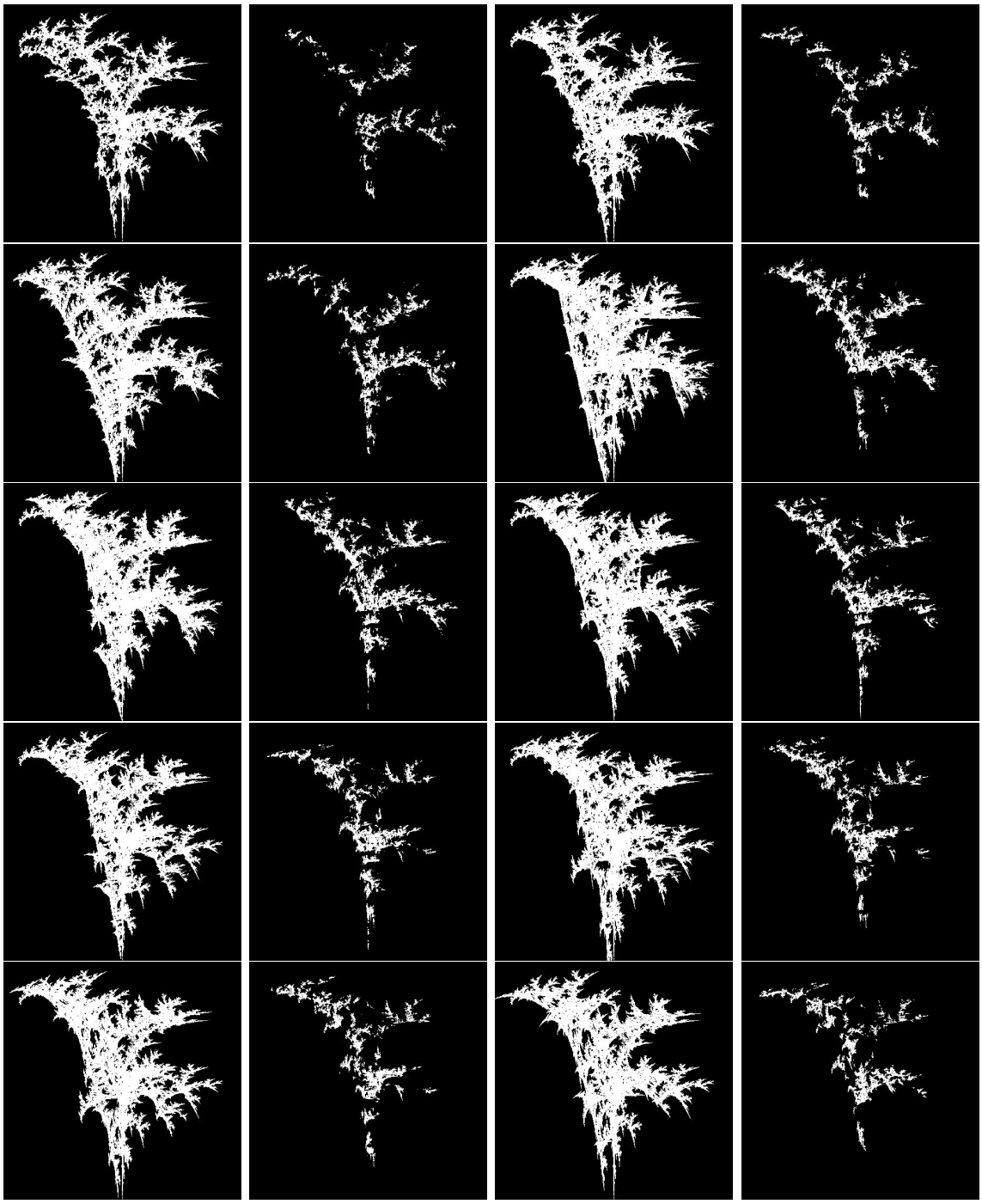

Figure 9: (l-r, t-b) Union (left) and intersection (right) sets of the input and approximating fractal images of the `bush` example for 5,000 to 7,250 iterations (step-size 250).

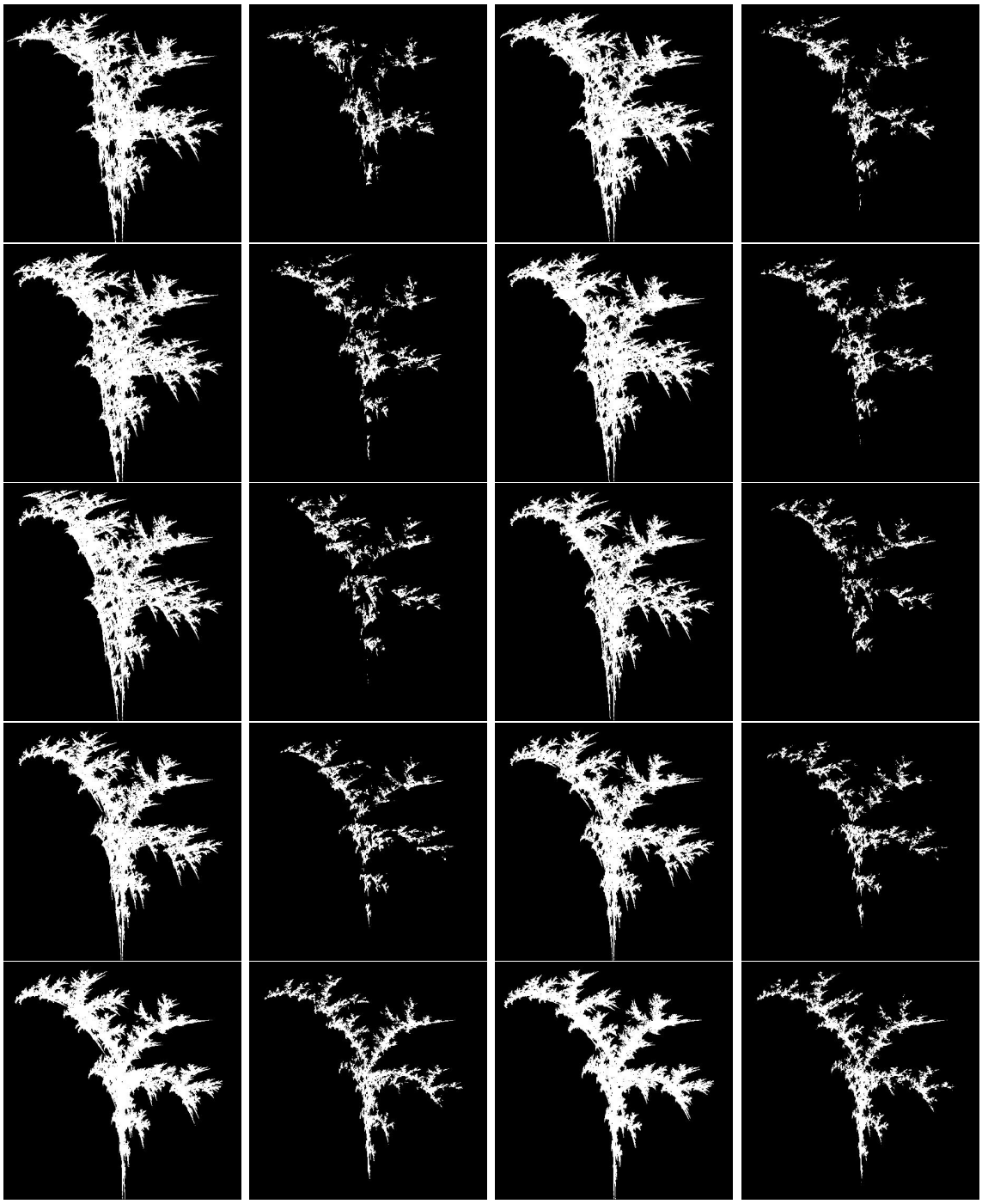

Figure 10: (l-r, t-b) Union (left) and intersection (right) sets of the input and the approximating fractal images of the `bush` example for 7,500 to 9,750 iterations (step-size 250).

6.2.2. Numerical results

The good graphical results described in previous paragraphs are well supported by the numerical results. Tables 2 and 3summarize the main results obtained for the `bush` and the `blocks` fractal examples, respectively. The tables show the results obtained for the global best solution for the 10,000 generations sampled with step-size 250 (in rows). For each generation value g within this range, the following data are reported (in columns): number of generation, g; number of contractive maps, η, of the best solution at generation g; active (black) and background (white) pixels of the reconstructed image (labelled as $\mathcal{R}$ onwards), denoted as $\blacksquare_{\mathcal{R}}$ and $\square_{\mathcal{R}}$, respectively; number of pixels with different binary values for the input image, $\mathcal{I}$ and the reconstructed image, $\mathcal{R}$, denoted as $\blacksquare_{\mathcal{I}\neq\mathcal{R}}$; value of the Hamming similarity function (see Eq. (10)), $\mathcal{S}_\Delta$ (between 0 and 1; the higher, the better); number of pixels in the intersection and union sets $\mathcal{I}\cap\mathcal{R}$ and $\mathcal{I}\cup\mathcal{R}$, denoted as $\blacksquare_{\cap}$ and $\blacksquare_{\cup}$, respectively; value of the intersection similarity function (see Eq. (11)), $\mathcal{S}_\cap$; and finally, two new metrics for this paper, the rate of active pixels in the intersection set $\mathcal{I}\cap\mathcal{R}$ with respect to the active pixels of the input image, $\mathcal{I}$, and the reconstructed image, $\mathcal{R}$, denoted as $\blacksquare_{\cap}/\blacksquare_{\mathcal{I}}$ and $\blacksquare_{\cap}/\blacksquare_{\mathcal{R}}$, respectively.

Table 2 provides a lot of information to explain the previous observations about the graphical results. For instance, the values of η in second column show that indeed the method can change automatically the number of contractive maps over the time in order to get a better approximation of the input image. These changes do not necessarily lead to an improvement of the similarity between $\mathcal{I}$ and $\mathcal{R}$ at the short term. For instance, the method changed from $\eta = 8$ at generation $g = 2,000$ to $\eta = 5$ at generation $g = 2,250$, even although $\mathcal{S}_\Delta$ decreases from 0.850405 to 0.845353. A similar effect occurs at generations $5,000$, $6,500$ and others. These situations are allowed in order to avoid the method to get stuck in a local optimum and also to explore the search space more efficiently looking for more promising solutions. This is also the reason why we also removed the elitism of the previous conference paper in this enhanced version of the algorithm. Note also that the value of η in the table changes from 5 to 10 over the generations (the ground truth for the input image is $\eta = 5$, but this knowledge is never used in the method), even although we allow it to take values between 3 and 12 (actually, these values are used for the initial random population, and they survive for a while, but the table only reports the best solution of the population). This means that the method is able to automatically select suitable values for η within a smaller subset of the initial proposal. We also remark that the value for the global best is never below 5, a clear indication that the input image cannot be replicated well with fewer than 5 contractive maps. Finally, we remark that the value of η keeps constant for the last 1,500 iterations, once convergence is achieved. Furthermore, this value matches the real value of the input image, a clear evidence of the strong ability of our method for fractal image reconstruction. A similar behavior can be observed in Table 3, although even larger values of η can be obtained for this example. This effect can be explained by the fact that this fractal image covers a larger area than the other example, so the method initially assigns more contractive

Table 2: Numerical results of our method for the `bush` fractal example (see the main text for details).

g	η	$\blacksquare_{\mathcal{R}}$	$\square_{\mathcal{R}}$	$\square_{\mathcal{I}\neq\mathcal{R}}$	$\mathcal{S}_{\Delta}$	$\blacksquare_{\cap}$	$\blacksquare_{\cup}$	$\mathcal{S}_{\cap}$	$\blacksquare_{\cap}/\blacksquare_{\mathcal{I}}$	$\blacksquare_{\cap}/\blacksquare_{\mathcal{R}}$
0	7	76,876	125,624	75,304	0.628128	11,681	86,985	0.134288	0.536072	0.151946
250	7	34,502	167,998	41,602	0.794558	7,345	48,947	0.150060	0.337081	0.212886
500	8	35,791	166,709	42,223	0.791491	7,679	49,902	0.153882	0.352409	0.214551
750	8	29,185	173,315	35,713	0.823640	7,631	43,344	0.176057	0.350207	0.261470
1,000	6	23,466	179,034	32,600	0.839012	6,328	38,928	0.162557	0.290408	0.269667
1,250	7	28,676	173,824	30,898	0.847417	9,784	40,682	0.240499	0.449013	0.341191
1,500	7	31,412	171,088	29,408	0.854775	11,897	41,305	0.288028	0.545984	0.378741
1,750	7	29,544	172,956	26,860	0.867358	12,237	39,097	0.312991	0.561588	0.414196
2,000	8	17,613	184,887	30,293	0.850405	4,555	34,848	0.130711	0.209041	0.258616
2,250	5	25,378	177,122	31,316	0.845353	7,926	39,242	0.201977	0.363745	0.312318
2,500	7	21,136	181,364	31,512	0.844385	5,707	37,219	0.153336	0.261909	0.270013
2,750	7	26,626	175,874	33,278	0.835664	7,569	40,847	0.185301	0.347361	0.284271
3,000	7	33,603	168,897	35,061	0.826859	10,166	45,227	0.224777	0.466544	0.302533
3,250	6	22,836	179,664	30,792	0.847941	6,917	37,709	0.183431	0.317439	0.302899
3,500	6	20,497	182,003	30,333	0.850207	5,977	36,310	0.164610	0.274300	0.291604
3,750	7	25,409	177,091	32,543	0.839294	7,328	39,871	0.183793	0.336301	0.288402
4,000	7	24,821	177,679	28,069	0.861388	9,271	37,340	0.248286	0.425470	0.373514
4,250	8	19,401	183,099	26,245	0.870395	7,473	33,718	0.221632	0.342955	0.385186
4,500	9	19,923	182,577	25,383	0.874652	8,165	33,548	0.243383	0.374713	0.409828
4,750	9	21,545	180,955	28,811	0.857723	7,262	36,073	0.201314	0.333272	0.337062
5,000	9	17,163	185,337	28,927	0.857151	5,013	33,940	0.147702	0.230060	0.292082
5,250	10	24,647	177,853	31,997	0.841991	7,220	39,217	0.184104	0.331345	0.292936
5,500	9	31,957	170,543	35,299	0.825684	9,224	44,523	0.207174	0.423313	0.288638
5,750	10	36,942	165,558	35,376	0.825304	11,678	47,054	0.248183	0.535934	0.316117
6,000	10	35,116	167,384	31,182	0.846015	12,862	44,044	0.292026	0.590271	0.366272
6,250	10	30,230	172,270	29,804	0.85282	11,108	40,912	0.271512	0.509775	0.367450
6,500	10	25,871	176,629	29,545	0.854099	9,058	38,603	0.234645	0.415695	0.350122
6,750	9	27,978	174,522	31,604	0.843931	9,082	40,686	0.223222	0.416797	0.324612
7,000	8	28,649	173,851	33,277	0.835669	8,581	41,858	0.205003	0.393804	0.299522
7,250	8	24,790	177,710	30,698	0.848405	7,941	38,639	0.205518	0.364433	0.320331
7,500	7	22,282	180,218	27,716	0.863131	8,178	35,894	0.227838	0.375311	0.367023
7,750	6	23,972	178,528	28,344	0.860030	8,709	37,053	0.235042	0.399679	0.363299
8,000	6	25,982	176,518	29,772	0.852978	9,000	38,772	0.232126	0.413034	0.346394
8,250	6	23,415	179,085	26,607	0.868607	9,299	35,906	0.258982	0.426755	0.397139
8,500	5	23,079	179,421	28,999	0.856795	7,935	36,934	0.214843	0.364158	0.343819
8,750	5	15,578	186,922	24,066	0.881156	6,651	30,717	0.216525	0.305232	0.426948
9,000	5	17,144	185,356	21,854	0.892079	8,540	30,394	0.280977	0.391923	0.498133
9,250	5	17,599	184,901	22,087	0.890928	8,651	30,738	0.281443	0.397017	0.491562
9,500	5	23,433	179,067	16,475	0.918642	14,374	30,849	0.465947	0.659662	0.613408
9,750	5	23,123	179,377	14,843	0.926701	15,035	29,878	0.503213	0.689995	0.650218
10,000	5	23,118	179,382	14,840	0.926716	15,040	29,862	0.503650	0.690225	0.650575

Table 3: Numerical results of our method for the `blocks` fractal example (see the main text for details).

g	η	$\blacksquare_{\mathcal{R}}$	$\square_{\mathcal{R}}$	$\square_{\mathcal{I}\neq\mathcal{R}}$	$\mathcal{S}_{\Delta}$	$\blacksquare_{\cap}$	$\blacksquare_{\cup}$	$\mathcal{S}_{\cap}$	$\blacksquare_{\cap}/\blacksquare_{\mathcal{I}}$	$\blacksquare_{\cap}/\blacksquare_{\mathcal{R}}$
0	6	71,738	130,762	75,850	0.625432	33,111	108,961	0.303879	0.470768	0.461555
250	5	39,994	162,506	71,276	0.648020	19,526	90,802	0.215039	0.277618	0.488223
500	6	41,041	161,459	72,555	0.641704	19,410	91,965	0.211059	0.275969	0.472942
750	7	43,749	158,751	74,477	0.632212	19,803	94,280	0.210045	0.281557	0.452650
1,000	7	47,134	155,366	71,342	0.647694	23,063	94,405	0.244299	0.327907	0.489307
1,250	8	53,603	148,897	70,847	0.650138	26,545	97,392	0.272558	0.377413	0.495215
1,500	8	63,913	138,587	67,871	0.664835	33,188	101,059	0.328402	0.471863	0.519268
1,750	9	74,864	127,636	67,278	0.667763	38,960	106,238	0.366724	0.553928	0.520410
2,000	9	77,051	125,449	68,157	0.663422	39,614	107,771	0.367576	0.563227	0.514127
2,250	9	61,484	141,016	68,472	0.661867	31,673	100,145	0.316271	0.450323	0.515142
2,500	8	61,124	141,376	67,018	0.669047	32,220	99,238	0.324674	0.458101	0.527125
2,750	8	60,638	141,862	67,048	0.668899	31,962	99,010	0.322816	0.454432	0.527095
3,000	9	65,275	137,225	65,867	0.674731	34,871	100,738	0.346155	0.495792	0.534217
3,250	10	73,025	129,475	64,299	0.682474	39,530	103,829	0.380722	0.562033	0.541321
3,500	7	58,583	143,917	64,765	0.680173	32,076	96,841	0.331223	0.456053	0.547531
3,750	8	65,465	137,035	64,379	0.682079	35,710	100,089	0.356782	0.507720	0.545482
4,000	8	66,643	135,857	62,323	0.692232	37,327	99,650	0.374581	0.530711	0.560104
4,250	9	63,706	138,794	62,606	0.690835	35,717	98,323	0.363262	0.507820	0.560654
4,500	9	64,024	138,476	61,844	0.694598	36,257	98,101	0.369588	0.515497	0.566303
4,750	9	72,818	129,682	63,066	0.688563	40,043	103,109	0.388356	0.569326	0.549905
5,000	10	78,712	123,788	61,616	0.695723	43,715	105,331	0.415025	0.621534	0.555379
5,250	111	83,355	119,145	64,265	0.682642	44,712	108,977	0.410288	0.635710	0.536405
5,500	9	75,991	126,509	59,505	0.706148	43,410	102,915	0.421804	0.617198	0.571252
5,750	12	87,587	114,913	64,989	0.679067	46,466	111,455	0.416904	0.660648	0.530513
6,000	10	81,388	121,112	61,344	0.697067	45,189	106,533	0.424178	0.642492	0.555229
6,250	11	82,167	120,333	60,939	0.699067	45,781	106,720	0.428982	0.650909	0.557170
6,500	7	63,364	139,136	58,360	0.711802	37,669	96,029	0.392267	0.535573	0.594486
6,750	6	62,371	140,129	57,395	0.716568	37,655	95,050	0.396160	0.535374	0.603726
7,000	7	61,304	141,196	56,632	0.720336	37,503	94,135	0.398396	0.533213	0.611755
7,250	6	50,287	152,213	60,505	0.701210	30,058	90,563	0.331902	0.427361	0.597729
7,500	6	46,681	155,819	59,951	0.703946	28,532	88,483	0.322457	0.405664	0.611212
7,750	6	64,065	138,435	46,317	0.771274	44,041	90,358	0.487406	0.626169	0.687442
8,000	5	64,964	137,536	47,290	0.766469	44,004	91,294	0.482003	0.625643	0.677360
8,250	6	54,287	148,213	51,699	0.744696	36,461	88,160	0.413578	0.518398	0.671634
8,500	6	67,205	135,295	41,465	0.795235	48,037	89,502	0.536714	0.682984	0.714783
8,750	6	67,349	135,151	44,365	0.780914	46,659	91,024	0.512601	0.663392	0.692794
9,000	6	73,323	129,177	39,645	0.804222	52,006	91,651	0.567435	0.739415	0.709273
9,250	5	70,349	132,151	42,199	0.791610	49,242	91,441	0.538511	0.700117	0.699967
9,500	5	70,081	132,419	36,903	0.817763	51,756	88,659	0.583765	0.735860	0.738517
9,750	5	69,203	133,297	33,907	0.832558	52,815	86,722	0.609015	0.750917	0.763189
10,000	5	70,814	131,686	28,482	0.859348	56,333	84,815	0.664187	0.800936	0.795507

maps during the exploration phase. This number is then refined at later stages, where the number of contractive functions is reduced but their contractivity factors increase in order to compensate for the missing functions.

The number of black (white) pixels of the reconstructed image, given by $\blacksquare_{\mathcal{R}}$ ($\square_{\mathcal{R}}$) is a good indicator of the performance of the method. Obviously, this number should ideally match the value for the input image, $\blacksquare_{\mathcal{I}} = 21,790$ and $\blacksquare_{\mathcal{I}} = 70,334$ for the `bush` and `blocks` images, respectively. From third column in Tables 2 and 3, we can see that this value oscillates dynamically above and below of this value, until settling in the values $\blacksquare_{\mathcal{R}} = 23,118$ and $\blacksquare_{\mathcal{R}} = 70,814$, an error less than 6% and 1%, respectively.

Another valuable indicator is the number of pixels (either black or white) with different binary values for $\mathcal{I}$ and $\mathcal{R}$, $\blacksquare_{\mathcal{I} \neq \mathcal{R}}$, reported in the fifth column of the tables. This value is quite large at initial generation, but decreases over the generations, drastically at the beginning and at slower pace at late stages, until reaching the plateau value at $\blacksquare_{\mathcal{I} \neq \mathcal{R}} = 14,840$ for the `bush` example in Table 2. From this amount, 6,754 pixels correspond to the difference set $\mathcal{I} - \mathcal{R}$, while 8,086 belong to the difference set $\mathcal{R} - \mathcal{I}$. The value of $\blacksquare_{\mathcal{I} \neq \mathcal{R}}$ is used to compute the Hamming similarity function, one of the best indicators of the quality of the approximation. Its values are shown in the sixth column of the table. Note that the final reconstructed image has a Hamming similarity of 0.926716, an excellent rate of matching of about 92.6%. A similar behavior is observed for the blocks example in Table 3, with a final Hamming similarity of 0.859348, a rate of matching of about 85.9%. To our knowledge, no other previous method reported values of this order, even although some methods consider much simpler images than those used in our benchmark.

The number of black pixels in the intersection $\mathcal{I} \cap \mathcal{R}$ and union $\mathcal{I} \cap \mathcal{R}$ sets is very useful to quantify the degree of similarity of both images (the reader can see the evolution of the union and the intersection sets for the 10,000 generations with step-size 250 in to additional accompanying *QuickTime* videos: *Video3.mov* (length: 82 seconds; size: 1.5 MB) and *Video4.mov* (length: 82 seconds; size: 0.8 MB). These values, reported in columns 7-8 of the table, are used to compute the intersection similarity function $\mathcal{S}_{\cap}$, shown in column 9. Note that this value oscillates up and down until reaching a final value of $\mathcal{S}_{\cap} = 0.503650$. This value might appear surprising in the light of the very good matching between $\mathcal{I}$ and $\mathcal{R}$, confirmed by our graphical results and other indicators such as $\mathcal{S}_{\Delta}$. However, it should be taken into account that any minor distortion of the image (e.g., rotation, scaling, or translation) might induce substantial changes on the values of this similarity function, even although the general shape of the image is still well reproduced. Furthermore, even if these variations occur at a local level, they are amplified by the self-similar nature of the fractal. As a result, they have a dramatic effect on the numerical results. On the other hand, we point out that this metric talks about the black pixels exclusively. As a result, it can also be strongly affected by the size of the fractal image. To analyze this effect in detail, we compute $\blacksquare_{\cap}/\blacksquare_{\mathcal{I}}$ and $\blacksquare_{\cap}/\blacksquare_{\mathcal{R}}$, in columns 10 and 11, respectively. Their final results indicate that the rate of overlapping between

Table 4: Comparative results of $\mathcal{S}_\Delta$ for the `blocks` and `bush` examples with our method and other alternative approaches (best results in bold).

Method	`blocks` example	`bush` example
Multilayer perceptron [21]:	0.2154	0.3959
Simulated annealing [25]:	0.2671	0.3874
Genetic algorithms [22]:	0.5389	0.6087
Firefly algorithm [41]:	0.4613	0.5389
Original bat algorithm [42]:	0.6245	0.6874
Modified bat algorithm w/o LS [15]:	0.6693	0.7385
Modified bat algorithm with LS [15]:	0.6804	0.7744
Modified OFS-RDS bat algorithm:	**0.8617**	**0.9267**

$\mathcal{I} \cap \mathcal{R}$ and $\mathcal{I}$ (resp. $\mathcal{R}$) is about 69% (resp. 65%) for the black pixels. These results seem to disagree our previous conclusion of the good matching between both images. However, a further analysis show that the similar computations for the white pixels, i.e. $\square_\cap/\square_\mathcal{I}$ and $\square_\cap/\square_\mathcal{R}$, show rates of 95.52% and 96.23%, respectively. This observation is supported by the results for the `blocks` image, which has a significant larger amount of black pixels. In this case, the rates $\blacksquare_\cap/\blacksquare_\mathcal{I}$ and $\blacksquare_\cap/\blacksquare_\mathcal{R}$, are about 80% and 79.5%, respectively, leading to a better value for the intersection similarity function: $\mathcal{S}_\cap = 0.664187$. As a conclusion, when the number of black pixels of the image is small compared to the number of white pixels, the $\mathcal{S}_\cap$ metric can be somehow misleading, and it should be complemented with (or even replaced by) other more reliable indicators, particularly $\mathcal{S}_\Delta$.

6.3. Comparative Analysis

It is always advisable to carry out a comparative work of the proposed method with other alternative approaches described in the literature. To this aim, seven different methods are considered: artificial neural networks, simulated annealing, genetic algorithms, the firefly algorithm, the original bat algorithm, and two variants of the the modified bat algorithm introduced in our previous conference paper [15], namely without and with local search (LS). They are good representatives of different families of methods: neural networks is one of the most widely used artificial intelligence methodologies; simulated annealing is a popular single-particle method; genetic algorithms are a standard method in the field of evolutionary computation; and firefly and bat algorithms are popular choices of population-based swarm intelligence algorithms. Furthermore, all these set of methods have already been applied to this problem.

Regarding the configuration and parameter tuning of these alternative methods, for the neural networks we consider a multilayer perceptron (MLP), which is reported to be a universal function approximator. In our configuration, the MLP includes 30 neurons

(as many as the actual free variables of the problem) in a single hidden layer, with the Levenberg–Marquardt back propagation algorithm used for training [26, 29]. Whenever possible, we consider a similar parameter tuning as in this paper for a fair comparison of the methods. For instance, we consider a population of 100 individuals and $10,000$ iterations for the genetic algorithms, firefly algorithm, and bat algorithm, while a total of 10^6 iterations are considered for simulated annealing to compensate the fact that only a single particle is considered. In this way, we consider the same number of function evaluations as with the other methods.

Table 4 shows the comparative results for the `blocks` and `bush` examples (arranged in columns). The different methods used in the comparison are listed in rows. For each method, the value of $\mathcal{S}_\Delta$ is reported. The best results are highlighted in bold for easier identification. From the table, we can see that the method in this paper outperforms all other alternative approaches in this comparison. Furthermore, the improvement rate is really significant, not merely incremental. These results are a good validation of this method with respect to the current state-of-the-art methods in the field.

6.4. Computational Complexity and CPU Times

It is well-known that it is not possible to determine the computational complexity of metaheuristic methods (such as the bat algorithm used in this paper) on a general basis, as it depends on a number of factors such as the population size, number of generations, number of free variables, parameter tuning, landscape of the search space, and so on. To make things even harder, the metaheuristic methods cannot always guarantee to find the global optimum. In this situation, the classical approach is to compute either the number of functions evaluations or the CPU time of the algorithm. For the examples in this paper and the parameter tuning described in Sect. 6.1, the CPU time for a single execution is about 5–8 hours, depending on the size and complexity of the image. These CPU times have been obtained with *Matlab*, version 2018a running on a personal PC with a 3.7 GHz. Intel Core i7 processor and 16 GB. of RAM. Obviously, these CPU times make the method unsuitable for real-time applications. However, they are quite competitive with respect to other similar approaches, such as those discussed in our comparative work. Regarding the complexity of the similarity functions, they have a worst case time complexity of $O(M \times N)$ for binary images of size $M \times N$. As a result, their complexity is linear with the number of pixels of the image.

7. Conclusions and Future Work

This paper presents a new method for fractal compression of bitmap binary images encoded as IFS. The method is based on the application of a modified bat algorithm to compute all the parameters of the IFS code of the image automatically. This work follows up our previous paper in [15] extended and enhanced through three new valuable features:

1. The bat algorithm in the previous conference paper is replaced by an improved version based on the optimal forage strategy (OFS) and the random disturbance strategy (RDS). These two strategies improve the search capability of the method, by promoting large benefit moves during the local search phase without restricting them to the neighborhood of the current best, and by allowing some negative moves, with the goal to prevent the method to get stuck in local optima. The method also includes a switching procedure for a better balance between early exploration and late intensification.
2. Opposed to the conference paper, the number of contractive maps, η, is allowed to change for different individuals in the population and also to change dynamically over the iterations. Our new method computes the optimal value of η automatically and accurately.
3. This work considers several similarity functions and other metrics to improve our understanding of the method and improve its accuracy.

This new method is applied to a benchmark of two binary fractal images exhibiting a complex and irregular fractal shape. The graphical and numerical results show that the method performs very well, being able to reconstruct the input images with a Hamming similarity percentage of 86% and 92%, much better than the results obtained by previous approaches. From these results, we conclude that the method is really promising and has a lot of potential in the field.

Of course, the method has also some limitations. Perhaps the most critical one concerns the computation times, which ranges about 4–10 hours for the experiments described here and others not reported here to keep the paper in manageable size. These CPU times are prohibitive for applications requiring high-speed encoding. On the contrary, the decoding time is extremely fast and actually well suited for real-time applications. On the other hand, the accuracy might be still further improved, at least, theoretically. Although it is not realistic to expect a perfect matching, we think that the number of mismatches between the input and the reconstructed images might be reduced even a little bit more. We are currently working to improve these features.

Other ideas for future work in the field include the extension of this method to general black-and-white images containing shades of gray, the development of methods for effective IFS encoding of color images for different color spaces, and the development of new approaches to reduce the computational load in order to make this technology efficient for image and video storage with regards to video streaming and other online applications.

References

[1] Abenda, S., Demko, S., Turchetti, G.: Local moments and inverse problem for fractal measures. *Inv. Probl.*, **8**, 739–750 (1992).

[2] Barnsley, M.F., Demko, S.: Iterated function systems and the global construction of fractals. *Proceedings Royal Society of London*, **A399**, 243–275 (1985).

[3] Barnsley, M.F., Ervin, V., Hardin, D., Lancaster, J.: Solution of an inverse problem for fractal and other sets. *Proceedings of the National Academy of Science*, **83**, 1975–1977 (1986).

[4] Barnsley, M.F., Sloan, A.D.: A better way to compress images. *BYTE Magazine*, Jan. 1988 (1988).

[5] Barnsley, M.F.: *Fractals Everywhere (2nd Ed.)*. Academic Press, San Diego (1993).

[6] Barnsley, M.F., Hurd, L.P.: *Fractal Image Compression*. AK Peters/CRC Press (1993).

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

[8] Cai, X., Gao, X.Z., Xue, Y.: Improved bat algorithm with optimal forage strategy and random disturbance strategy. *Int. J. Bio-Inspired Computation*, **8**(4), 205–214 (2016).

[9] Dasgupta, D., Hernandez, G., Niño,, F.: An evolutionary algorithm for fractal coding of binary images. *IEEE Transactions on Evolutionary Computation*, **4**(2), 172–181 (2000).

[10] Elton, J.H.: An ergodic theorem for iterated maps. *Ergodic Theory Dynam. Syst.*, **7**, 481–488 (1987).

[11] Evans, A.K., Turner, M.J.: Specialisation of evolutionary algorithms and data structures for the IFS inverse problem. In: M.J. Turner (Ed.). *Proceedings of the Second IMA Conference on Image Processing: Mathematical Methods, Algorithms, and Applications* (1998).

[12] Falconer, K.: *Fractal Geometry: Mathematical Foundations and Applications (2nd Ed.)*. John Wiley & Sons, Chichester, England (2003).

[13] Fisher, Y. (ed.): *Fractal Image Compression: Theory and Applications*. Springer-Verlag, Berlin (1995).

[14] Forte B., Vrscay, E.R.: Solving the inverse problem for measures using iterated function systems: A new approach. *Adv. Appl. Prob.*, **27**, 800–820 (1995).

[15] Gálvez, A., Iglesias, A.: Modified bat algorithm with local search for fractal image compression of bitmap images. *Proc. of Int. Conf. on Cyberworlds, CW 2019*, IEEE Computer Society Press, Los Alamitos, CA, 199–206 (2019).

[16] Goentzel, B.: Fractal image compression with the genetic algorithm. *Complexity International*, **1**, 111–126 (1994).

[17] Gonzalez, R.C., Woods, R.E.: *Digital Image Processing. 4th Edition.* Pearson Prentice Hall, N.J. (2017).

[18] Graf, S.: Barnsley's scheme for the fractal encoding of images. *Journal of Complexity*, **8**, 72–78 (1992).

[19] Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A.: A multifractal analysis of IFSP invariant measures with application to fractal image generation. *Fractals* **4**(1), 17–27 (1996).

[20] Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A., Rodríguez, V.J.: Generating and rendering fractal images. *The Mathematica Journal*, **7**(1), 6–13 (1997).

[21] Haykin, S.: *Neural Networks: A Comprehensive Foundation (Second Edition).* Prentice Hall (1998).

[22] Holland, J.H.: *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press (1975).

[23] Hutchinson, J.E.: Fractals and self similarity. *Indiana Univ. Math. Journal*, **30**(5), 713–747 (1981).

[24] Jacquin, A.E.: Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, **1**(1), 18–30 (1992).

[25] Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science*, **220**(4598), 671–680 (1983).

[26] Levenberg, K.: A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics* **2**(2). 164–168 (1944).

[27] Lutton, E.,Véhel, J.L., Cretin, G., Glevarec, P., Roll, C.: Mixed IFS - resolution of the inverse problem using genetic programming. INRIA Rapport 2631 (1995).

[28] Luus, R., Jaakola, T.H.I.: Optimization by direct search and systematic reduction of the size of search region. *American Institute of Chemical Engineers Journal (AIChE)*, **19**(4) 760-766 (1973).

[29] Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, **11**(2), 431–441 (1963).

[30] Muruganandham, A., Wahida, R.S.D.: Adaptive fractal image compression using PSO. *Procedia Computer Science*, **1**, 338–344 (2010).

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

[32] Rabbani, M., Jones, P.W.: *Digital Image Compression Techniques*. SPIE Tutorial Texts in Optical Engineering, Vol. 7. SPIE Press (1991).

[33] Saupe, D., Hamzaoui, R.: A review of the fractal image compression literature. *Computer Graphics*, **28**(4), 268–276 (1994).

[34] Sayood, K.: *Introduction to Data Compression, 4th Edition*. Morgan Kaufmann (2012).

[35] 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).

[36] Tseng, C.C., Hsieh, J.G., Jeng, J.H.: Fractal image compression using visual-based particle swarm optimization. *Image and Vision Computing*, **26**, 1154–1162 (2008).

[37] Vyrscay, E.R.: Moment and collage methods for the inverse problem of fractal construction with iterated function systems, In Peitgen, H.O. et al. (editors): *Fractals in the Fundamental and Applied Sciences*. Elsevier (1991).

[38] Wadstromer, N.: An approach to the inverse IFS problem using the Kantorovich metric. *Fractals*, **5**(1), 89–99 (1997).

[39] Wu, M.S., Teng, W.C., Jeng, J.H., Hsieh, J.G.: Spatial correlation genetic algorithm for fractal image compression, *Chaos, Solitons and Fractals*, **28**(2), 497–510 (2006).

[40] Wu, M.S., Jeng, J.H., Hsieh, J.G.: Schema genetic algorithm for fractal image compression, *Engineering Applications of Artificial Intelligence*, **20**, 531–538 (2007).

[41] Yang, X.S.: Firefly algorithms for multimodal optimization. *Lectures Notes in Computer Science*, **5792**, 169–178 (2009).

[42] Yang, X.S.: A new metaheuristic bat-inspired algorithm. *Studies in Computational Intelligence*, **284**, 65–74 (2010).

[43] Yuan, W.X., Ping, L.F., Guo, W.S.: Fractal image compression based on spatial correlation and hybrid genetic algorithm. *J. Vis. Commun. Image R.*, **20**, 505–510 (2009).

[44] Zheng, Y., Liu, G.R., Niu, X.X.: An improved fractal image compression approach by using iterated function system and genetic algorithm. *Computers and Mathematics with Applications*, **51**, 1727–1740 (2006).