Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08954-7

S.I. : 2021 INDIA INTL. CONGRESS ON COMPUTATIONAL INTELLIGENCE

Hybrid GA-PSO method with local search and image clustering for automatic IFS image reconstruction of fractal colored images

Akemi Gálvez[1,2] · Iztok Fister[1,3] · Suash Deb[4,5] · Iztok Fister Jr.[3] · Andrés Iglesias[1,2]

Received: 5 September 2022 / Accepted: 21 July 2023

Abstract
Over the past few decades, the application of iterated function systems (IFS) in reconstructing fractal images has been a challenging research area. Numerous methods have been proposed to address this issue. However, they generally focus on binary or grayscale images, neglecting the color component of the process. Consequently, they are unsuitable for reconstructing colored images. In a previous paper presented at the ISCMI 2021 conference, the authors introduced a novel approach that utilizes the cuckoo search algorithm and *k*-means clustering for IFS fractal reconstruction of colored images. Building upon that work, this paper introduces an enhanced and extended method by combining genetic algorithms (GAs) and particle swarm optimization (PSO) with local search and image clustering. In this approach, GA and PSO are mutually coupled to automatically determine the color of the contractive functions and the IFS parameters, respectively. The output of each method serves as the input for the other in an iterative manner. Main contributions of this method are: (1) it computes automatically the optimal number and IFS code of the contractive functions; (2) the color of the contractive functions is determined automatically through an optimization process using GA; (3) a local refinement step is performed to further enhance the final solution. Overall, this new method yields highly accurate results in reconstructing the geometry and color of input fractal images, without requiring any additional information about the target beyond the bitmap image.

Keywords Swarm intelligence · Genetic algorithms · Particle swarm optimization · Image reconstruction · Color fractal images · Iterated function systems · Collage theorem

1 Introduction

1.1 Motivation

Fractals are fascinating mathematical objects, typically showing complicated geometrical patterns when drawn on the screen. Strikingly, such complex geometric shapes are generally obtained through very simple production rules. There are several ways to compute and display fractal objects. Popular procedures include escape-time fractals, recursive fractals, finite subdivision rules, random processes (e.g., Brownian motion, percolation clusters, diffusion-limited aggregation, Lévy flights, random walks), strange attractors of dynamical systems, L-systems, and others. The interested reader is referred to [1, 2] for a general overview on the field.

One of the most popular methods for fractal image generation is based on the so-called *Iterated Function Systems* (IFS). Basically, an IFS is a finite set of contractive affine maps $\{\varphi_i\}_{i=1,\ldots,N}$, defined on a complete metric

✉ Andrés Iglesias
iglesias@unican.es

1 Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avda. de los Castros, s/n, 39005 Santander, Spain

2 Faculty of Pharmaceutical Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan

3 Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroska cesta 34, 2000 Maribor, Slovenia

4 IT & Educational Consultant, Ranchi, India

5 Distinguished Professorial Associate, Decision Sciences and Modeling Program, Victoria University, Melbourne, Australia

Published online: 04 November 2023

space, $\mathcal{M}$ [3, 4]. These affine maps depend on several parameters encapsulating different 2D geometrical operations (scaling, rotation, translation). The set of parameter values for all affine maps of the IFS is called the IFS code. It can be proved that every IFS has a unique compact fixed set, $\mathcal{A}$, usually known as the attractor of the IFS. Interestingly, the graphical representation of this attractor is a fractal image.

The use of IFS goes beyond the generation of beautiful fractal images. Actually, one of the most exciting applications of the IFS appears in image reconstruction. Given a digital image, it is possible to obtain an IFS whose attractor replicates the input image accurately. Amazingly, this input image does not necessarily need to be a fractal image; it can be *any* image. This is a very surprising result, even shocking at first sight, which is based on the observation 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 [1]. In this way, the central task of fractal image reconstruction 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 [5, 6].

Although labeled as a very promising technique, the IFS image reconstruction is severely affected by a limiting factor: given a 2D input image, it is very difficult to obtain the IFS code (i.e., the parametric values of the IFS) providing a close approximation of the input image. This problem, usually referred to as *IFS fractal image reconstruction* problem, has become more challenging than expected. Although several approaches have been published during the last decades to address this problem (see Sect. 2 for details), the field still lacks a general methodology solving this question in the general case.

An additional issue is that the current methods for IFS image reconstruction address the problem from a purely geometric standpoint. As a result, they can only be applied to binary images, while the case of colored images is not actually considered. A previous paper presented by the authors at the ISCMI 2021 conference [7] addressed the reconstruction of colored images with IFS by introducing the color as a new free variable vector. The method combined the cuckoo search algorithm and k-means clustering to perform reconstruction of both the geometry and the color of the fractal image. However, the method had several limitations. In this paper, the method introduced in our previous conference paper in [7] is extended and enhanced in several ways. The main contributions of this work can be summarized as follows:

- In the previous conference paper, the number of contractive functions of the IFS was assumed to be known and given as an input of the method. Instead, in this paper this number is assumed to be unknown and computed in a fully automatic way.
- In the previous paper, the geometry and the color were not specialized, but encapsulated into a single representation. In this new method, genetic algorithms are used to address the discrete color subproblem, while particle swarm optimization is applied to address the continuous geometric subproblem, according to the actual (discrete or continuous) nature of the free variables of the general problem.
- In the previous paper, the geometry and the color were calculated through two different fitness functions, but their interplay was ignored. This limitation affected the accuracy of the results. In this new method, the interplay of the color and the geometry is embedded in the method by using a hybrid scheme, in which the genetic algorithms and the PSO do not work independently, but mutually coupled. Under this new strategy, the output of each method is used as input of the other in an iterative fashion. This new scheme takes into account the interplay of the geometry and color, as any new computation of one of them modifies automatically the results for the other.
- A new local refinement step through local search heuristics is now considered for further improvement of the final solution.
- The global fitness of the previous paper based on the arithmetic mean of the fitness functions for the geometry and the color is now replaced by a weighted convex combination of these individual fitness functions, and the best value for the weight is empirically determined.

As a consequence of these changes, the method provides much more accurate results than the previous method and in a fully automatic way (see Sect. 5 for details).

This paper is organized as follows: Section 2 discusses the previous work in the field. The mathematical background needed to follow the paper is presented in Sect. 3. The proposed method is explained in detail in Sect. 4. The obtained graphical and numerical results are discussed in Sect. 5. The paper closes with the main conclusions and some ideas for future work in the field.

2 Previous work

The notion of using fractals to encode digital images started with the seminal work by Michael Barnsley in the 1980s (see [5] for details). Based on earlier theoretical developments by Hutchinson in [4], he introduced the use

of fractal transformations to encode images in [8] and then in [9], where the collage theorem was firstly presented. That theorem showed that the digital images can be efficiently approximated through iterated function systems, leading to the first approaches and some patents for fractal image compression.

Interestingly, the graphical representation and encoding of digital images through IFS are a highly asymmetric problem. Rendering the attractor of a given IFS (usually called the direct problem) is a quite trivial task. Simple, fast, and efficient methods can be found, for instance, in [1, 3, 10]. On the contrary, the IFS encoding problem, i.e., obtaining an IFS whose attractor approximates a given digital image accurately (also known as the *inverse problem* and also as the *IFS image reconstruction problem*), is extremely difficult and elusive. Several methods have been described in the literature to address the inverse problem. First algorithms for fractal images were introduced in [9, 11]. These early methods were later extended to an automatic approach in [12], based on the concept of the partitioned iterated function systems (PIFSs). However, these methods rely on exhaustive search procedures, which are slow and computationally expensive. Several methods were proposed to alleviate this computational load, mostly based on quadtrees, rectangular partitions, and triangular partitions, in some cases combined with clustering. But because they are mostly based on greedy algorithms, they are still slow and generally exhibit very low efficiency. The list of these methods is quite large, and they are out of scope to be discussed here. The interested reader is referred to [13] for a detailed review. Other approaches include Gröbner basis [14], wavelet transform [15], gradient search [16], and moment matching [17]. Unfortunately, they are also computational expensive and can only be applied to some (generally simple) particular cases, while failing to solve the general problem.

It has been pointed out that the inverse problem can be formulated as an optimization problem. Consequently, there have been attempts to solve it using classical mathematical optimization techniques. However, it was noticed that these techniques are not powerful enough to solve the general problem. Then, the focus shifted toward metaheuristic techniques, which are generally applied to problems unsolvable with the classical approaches; for instance, when the objective function is not differentiable and hence gradient-based techniques can no longer be applied. Other cases include multimodal problems and ill-conditioned problems.

As a feasible alternative, nature-inspired population-based methods, such as those typically found in evolutionary computing and swarm intelligence, have been considered to tackle the IFS inverse problem. One of the most popular approaches was based on the application of evolutionary techniques, such as genetic algorithms [18, 19] and genetic programming [20, 21]. The combination of fractal compression using PIFS and genetic algorithms is reported in [22–24]. Similarly, an evolutionary algorithm has been applied in [25] for fractal coding of binary images. Other examples of these techniques can be found in [26, 27].

Swarm intelligence methods have also been applied in several papers. Fractal image compression with different variations of the particle swarm optimization can be found in [28, 29]. These works do not compute the IFS parameters, but instead perform exhaustive search of similarities between blocks of the image. The papers in [30] and [31] perform IFS image reconstruction through the bat algorithm and the firefly algorithm, respectively. Their results might be acceptable in some settings where the accuracy is not critical, but they provide suboptimal solutions. A more recent paper improves the accuracy while being able to compute the optimal number of contractive maps in addition to their parametric values [32].

All previous methods deal with the problem of shape reconstruction from a geometric point of view exclusively and can only be applied to binary (i.e., black and white) images. In contrast, the case of full-color images has been largely ignored in the literature. In a previous paper presented at the ISCMI 2021 conference, the authors introduced a new method to address the reconstruction of colored images with IFS by considering the color as a new free variable vector [7]. The method combined a swarm intelligence method called cuckoo search algorithm and *k*-means clustering to perform reconstruction of both the geometry and the color of the fractal image. An alternative method based on the bat algorithm and multilevel thresholding was also recently reported [33]. To the best of our knowledge, these are the only papers addressing the issue of IFS image reconstruction for colored images. However, these methods have also some limitations. In this paper, they are substantially extended and enhanced in several ways.

3 Mathematical background

This section provides the reader with the mathematical background needed to follow the paper. The interested reader is kindly referred to [3, 6, 34] for further details.

3.1 Contractive maps and iterated function systems

Let (X, d) be a metric space, where X is a non-empty set and d a distance defined on X. A *contractive map* φ on (X, d) is a function $\varphi : X \to X$ such that there is a real

number $0 \leq k < 1$ holding: $d(\varphi(x), \varphi(y)) \leq k.d(x, y)$, $\forall x, y \in X$. The *Banach fixed-point theorem* states that every contractive map φ on a non-empty complete metric space (X, d) has a unique fixed point, denoted as p. Moreover, given any $x \in X$, the sequence $x, \varphi(x), \varphi(\varphi(x)), \ldots$ obtained by iterative composition of φ with itself converges to the fixed point. In other words, if $x_{n+1} = \varphi(x_n)$, $\forall n \in \mathbb{N}$, then $\lim_{n\to\infty} x_n = p$.

Let $\mathcal{M} = (X, d)$ be a complete metric space. An *iterated function system (IFS)* is a finite set $\{\phi_i\}_{i=1,\ldots,N}$ of contractive maps $\phi_i : X \to X$ defined on $\mathcal{M}$. Generally, the IFS will be denoted as $\mathcal{W} = \{X; \phi_1, \ldots, \phi_N\}$. In this paper, we will focus on 2D bitmap images, so we consider onwards the complete metric space $(\mathbb{R}^2, d_2)$, where d_2 denotes the Euclidean distance. Therefore, the contractive affine maps ϕ_i are of the form:

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \phi_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \tag{1}$$

or equivalently, in matrix notation: $\mathbf{x}^* = \Phi_i(\mathbf{x}) = \mathbf{A}_i.\mathbf{x} + \mathbf{b}_i$, where $\mathbf{b}_i$ is a 2D vector describing the translations, and $\mathbf{A}_i$ is a 2×2 matrix encoding other 2D transformations, such as scaling and rotation. Since the maps ϕ_i are contractive, the matrix $\mathbf{A}_i$ has eigenvalues λ_1^i, λ_2^i such that $|\lambda_j^i| < 1$, $j = 1, 2$. Furthermore, the contractivity factor of the mapping ϕ_i, denoted as s_i, holds that $s_i = |det(\mathbf{A}_i)| < 1$, thus making the objects smaller after transformation.

From Eq. (1), we can see that any contractive affine map ϕ_i is uniquely defined by the set of parameters $(a_i, b_i, c_i, d_i, e_i, f_i)$. Furthermore, any IFS, $\mathcal{W}$, is fully characterized by the collection of parameters $\{(a_i, b_i, c_i, d_i, e_i, f_i)\}_{i=1,\ldots,N}$, called the *IFS code* of $\mathcal{W}$.

3.2 Hutchinson operator and IFS attractor

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

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

where $d_h(\mathcal{R}, \mathcal{S}) = \max_{x\in\mathcal{R}} \min_{y\in\mathcal{S}} d_2(x, y)$.

Because $(\mathbb{R}^2, d_2)$ is a complete metric space, it can be proved that $(\mathcal{C}_S(X), h)$ is also a complete metric space [3]. The *Hutchinson operator*, $\mathcal{H}$, defining the join action of all contractive maps ϕ_k on $\mathcal{C}_S(X)$, is given by:

$$\mathcal{H}(\mathcal{B}) = \bigcup_{k=1}^{N} \phi_k(\mathcal{B}) \qquad \forall \mathcal{B} \in \mathcal{C}_S(X) \tag{3}$$

Since all the ϕ_k are contractions in $(\mathbb{R}^2, d_2)$, $\mathcal{H}$ is also a contraction on $\mathcal{C}_S(X)$ with the induced Hausdorff metric [3, 4]. Then, the Banach fixed-point theorem states that $\mathcal{H}$ has a unique fixed point, $\mathcal{H}(\mathcal{A}) = \mathcal{A}$, called the *attractor of the IFS*. Interestingly, the attractor set $\mathcal{A}$ is a fractal image, regardless of the initial set $\mathcal{B}$ in Eq. (3).

There are several approaches to render the attractor of an IFS [35]. A simple and popular procedure is the *probabilistic algorithm*, where each contractive map ϕ_k is associated with a probability $\omega_k > 0$, such that $\sum_{k=1}^{N} \omega_k = 1$. The method proceeds iteratively: starting with an compact set $\mathcal{S}_0 \in \mathcal{C}_S(X)$, one of the maps of the IFS is randomly chosen with probability ω_k at iteration j to obtain $\mathcal{S}_j = \phi_k(\mathcal{S}_{j-1})$. The process is repeated again for the resulting set, and so on. It can be proved that $\overline{\{\mathcal{S}_j\}}_j = \mathcal{A}$, meaning that this iterative process can render the attractor [3, 10]. Theoretically the initial set $\mathcal{S}_0$ can be any compact set, but since the ϕ_κ are contractive, the size of $\mathcal{S}_0$ decreases over the iterations, eventually collapsing to a point. Consequently, it is advisable to take $\mathcal{S}_0$ as a single point for computational efficiency.

There are several approaches to compute the probabilities ω_κ [1, 36]. The most popular method, usually called *Barnsley's algorithm* (also, the *chaos game*), considers a probability value ω_κ related to the area filled by the contractive map ϕ_k, which is proportional to its contractive factor, $s_k = |det(\mathbf{A}_k)| = |a_k.d_k - b_k.c_k|$. Then, the method computes ω_i as:

$$\omega_i = \frac{s_i}{\sum_{k=1}^{N} s_k} \quad i = 1, \ldots, N. \tag{4}$$

In this paper, we follow this classical choice. Other methods are also feasible, even leading to more efficient values [10]. However, this problem is out of the scope of this work and is not addressed here.

3.3 The Collage theorem and the IFS reconstruction optimization problem

An important result in fractal theory is the *collage theorem* [9]. Given an IFS $\mathcal{W}$ on $(\mathbb{R}^2, d_2)$ with contractivity factor $s = \max_i s_i$, any $\mathcal{I} \in \mathcal{C}_S(\mathbb{R}^2)$, and a threshold $\epsilon \geq 0$, if $H\left(\mathcal{I}, \bigcup_{i=1}^{N} \phi_i(\mathcal{I})\right) \leq \varepsilon$, then $H(\mathcal{I}, \mathcal{A}) \leq \frac{1}{1-s} H\left(\mathcal{I}, \bigcup_{i=1}^{N} \phi_i(\mathcal{I})\right)$ with $H(.,.)$ being the induced Hausdorff metric on $\mathcal{C}_S(\mathbb{R}^2)$, and $\mathcal{A}$ the attractor of $\mathcal{W}$. In practical terms, this means that given any digital image $\mathcal{I}$, there exists an IFS whose attractor has a graphical representation $\mathcal{A}$ that approximates $\mathcal{I}$ accurately, according to the Hausdorff metric.

The challenging question is how to compute this IFS $\mathcal{W}$, which is the problem addressed in this paper. To

reconstruct a digital image $\mathcal{I}$, we need to compute the collection of parameters of an IFS (i.e., its IFS code) such that the attractor of the IFS, $\mathcal{H}(\mathcal{I})$, provides an accurate approximation of $\mathcal{I}$, according to a given metrics Δ. Thus, this problem can be formulated as the following optimization problem:

$$\underset{\{\mathbf{A}_i,\mathbf{b}_i\}_{i=1,\ldots,N}}{minimize} \left[\Delta\left(\mathcal{I}, \bigcup_{i=1}^{N} \phi_i(\mathcal{I})\right)\right] \tag{5}$$

such that $s_i = |det(\mathbf{A}_i)| < 1$ for all $i = 1, \ldots, N$.

It is worthwhile to remark that it is actually enough to approximate $\mathcal{I}$ by $\mathcal{H}(\mathcal{B})$, where $\mathcal{B}$ is *any* initial image (note that the attractor of the IFS $\mathcal{W}$ is independent of the initial image $\mathcal{B}$). Therefore, the optimization problem becomes:

$$\underset{\{\mathbf{A}_i,\mathbf{b}_i\}_{i=1,\ldots,N}}{minimize} \left[\Delta\left(\mathcal{I}, \bigcup_{i=1}^{N} \phi_i(\mathcal{B})\right)\right] \tag{6}$$

This problem is extremely challenging, as it is continuous (since all free variables in $\{\mathbf{A}_i, \mathbf{b}_i\}_{i=1,\ldots,N}$ are real-valued), constrained (because the corresponding functions ϕ_k have all to be contractive), and multimodal (there can be several global or local optima of the objective function Δ). In short, we have to solve a very difficult constrained multimodal continuous optimization problem. The problem is so challenging that it cannot be solved through classical mathematical optimization techniques. Several techniques have been proposed to tackle this issue, but only partial solutions have been reported in the literature so far (see Sect. 2 for details). Nowadays, the general problem still remains open and the community in the field is still searching for more powerful methods to address this problem. Furthermore, the difficulty of this problem is exacerbated by three additional major factors: (1) the optimal number of contractive functions, N is unknown in advance, so the method should typically include some procedure to determine it; (2) the problem can be high-dimensional for complex images, as they might require a large number of contractive maps for accurate reconstruction; (3) the optimization problem in Eq. (6) does not include any reference to color, so Eq. (5) must be modified to add this new attribute. This paper proposes a new method to solve all these problems simultaneously, as discussed in the next section.

4 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 individually in further detail.

4.1 Overview of the method

The input of our method is a fractal image, $\mathcal{I}$. It is assumed that $\mathcal{I}$ is given as a rectangular colored bitmap image of size $P \times Q$ (measured in pixels) on the compact domain $\Omega = [a, b] \times [c, d] \subset \mathbb{R}^2$. Without loss of generality, we can consider the domain to be the unit square, that is, $a = c = 0$, and $b = d = 1$. To be more precise, the image is mathematically represented by a two-dimensional matrix $\mathbf{D}$ of size $P \times Q$, where each matrix element $\mathbf{d}_{i,j}$ is a vector of three components, $\mathbf{d}_{i,j} = (R_{i,j}, G_{i,j}, B_{i,j})$, accounting, respectively, for the values of the red, green, and blue color channels of the corresponding pixel (i, j) of the image. According to the collage theorem in Sect. 3.3, there is an IFS $\mathcal{W}$, whose attractor approximates $\mathcal{I}$ accurately. Our goal is to reconstruct the input image $\mathcal{I}$ in both geometry and color using that IFS. Note that this task requires to obtain the optimal number of contractive functions, N, which was assumed to be known in our previous conference paper in [7], but it is automatically computed in this extended version. It also requires to determine all relevant parameters of the IFS $\mathcal{W}$, that is, its IFS code. Finally, it also requires to obtain the color map of the input figure. This color subproblem can be solved by determining the color of each of the N contractive maps of the IFS.

Figure 1 shows the workflow of our method. Given an input fractal image, the procedure consists of four main steps:

1. Firstly, color-based image clustering is carried out to identify the main regions in the image (see Sect. 4.2 for details).
2. Then, color histogram of each region of the cluster is computed and the dominant color for each cluster is extracted (see Sect. 4.3 for details).
3. Thirdly, a memetic approach hybridizing genetic algorithms and particle swarm optimization is applied to determine the optimal number of contractive functions, the IFS code and the color map of each contractive function via discrete and continuous optimization with genetic algorithms and particle swarm optimization (see Sect. 4.4 for details).
4. Finally, local search refinement is applied to enhance the optimal solution found in the previous step (see Sect. 4.5 for details).

The four steps of the method are individually discussed in detail in next sections. For a better description of the method, we consider an illustrative example of a colored fractal image. The image, called *Barnsley's fern* (or just *fern* for short), was first proposed by Michael Barnsley in the 1980s. It is one of the most iconic fractal images and is often used to illustrate the ability of IFS to generate organic shapes of high complexity. Although it consists of only

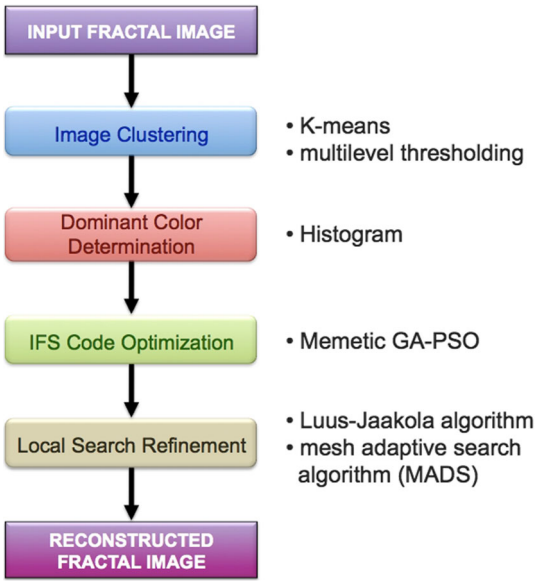


Fig. 1 Workflow of the method. On the left, the different steps of the method are shown (from top to bottom). On the right, the techniques used for each step are indicated

four contractive functions, the shape is challenging for geometric reconstruction, due to several factors: the disparity of areas covered by the contractive functions, positional rotations, and others. Therefore, it is a good benchmark to test the performance of the proposed method.

The fern image is displayed in Fig. 2 in two versions: full color on the left and the binary version on the right. The image has been generated with one million points following the rendering procedure explained in Sect. 3.2 and processed as a bitmap image of size 600×600 pixels. The final image consists of 360,000 pixels, encoded as a square matrix of order 600. The components of the matrix are color vectors in RGB color map for the colored image, and 0 s and 1 s for the binary image, corresponding to the pixels of the fractal and the pixels of the background, respectively. The number of active pixels (those in black in the binary version) is 80,240, giving a filling rate (i.e., percentage of active pixels over the total size of the image) of 22.28%.

4.2 Color-based image clustering of the input image

First step of our method is to perform color-based image clustering. The goal of this step is to identify the main regions (with respect to the color) of the input image and their dominant colors. In this paper, we consider two classical methods for image clustering for which several efficient implementations and computer libraries are available:

1. Similar to the previous conference paper in [7], we apply the well-known k-means method, which is based on minimization of within-cluster variances. In particular, k-means is based on minimizing the sum of the squared distances of each data to the cluster centroid. Thus, given a set of data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, the algorithm builds k clusters $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ as: $\min_{\Omega} E(\bar{g}_i) = \min_{\Omega} \sum_{i=1}^{k} \sum_{\mathbf{x_j} \in \mathbf{C_i}} ||\mathbf{x}_j - \bar{g}_i||^2$, where $\bar{g}_i$ denotes the centroid of cluster $\mathbf{C}_i$. The centroids are updated iteratively from the condition: $\frac{\partial E}{\partial \bar{g}_i} = 0$, yielding: $\bar{g}_i^{(t+1)} = \frac{1}{|\mathbf{C}_i^{(t)}|} \sum_{\mathbf{x_j} \in \mathbf{C_i^{(t)}}} \mathbf{x}_j$, where t indicates the iteration.
2. In addition, we consider a modification of the classical Otsu algorithm for automatic image binary thresholding [37], also based on the minimization of the intra-cluster intensity variance. Since in this paper we are dealing with colored images, we consider an implementation of the extension of the Otsu algorithm to the case of multi-level thresholding described in [38], which applies a recursive procedure accessing to a precomputed between-class variance through a look-up table.

These methods are applied to perform image clustering for different values of the number of clusters. The resulting clusters are stored to be subsequently used for comparative purposes between the initial fractal image and the reconstructed one throughout the iterative optimization during the next steps of our method.

It is interesting to remark that both algorithms, k-means and Otsu algorithm, are based on the same criterion, namely the minimization of the within-class variance. Their main difference is that Otsu method is an exhaustive global search algorithm, while k-means searches for the optimal solution locally. Figure 3 shows the image clustering results for the k-means method on the input colored

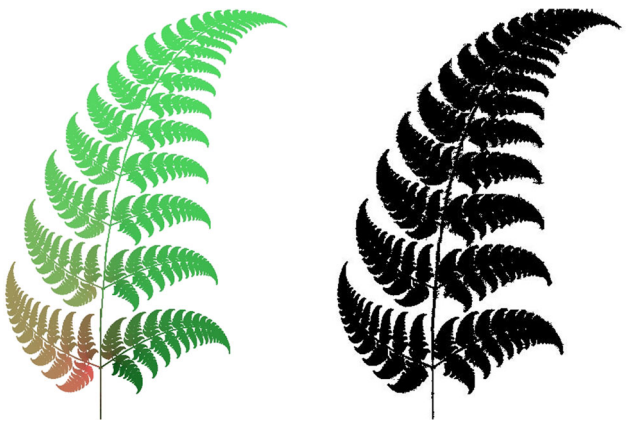


Fig. 2 Fractal colored image `fern` used in this paper in colored (left) and binary (right) versions

fractal image *fern* in Fig. 2 for different number of clusters ranging from 2 to 9 (including background). The results for image clustering with Otsu multi-level thresholding are very similar visually, so they are omitted here to keep the paper at reasonable length.

4.3 Dominant color determination for each cluster

Once the image clustering step is carried out, color histogram of each cluster is computed in order to determine its dominant color. To this aim, the pixels of each cluster are identified and their intensity values, described as triplets in RGB code, are stored and used to compute the histogram of the cluster. Then, the highest peak value in the histogram is selected to determine the dominant color of each cluster. The output of this step is a collection of vectors $\{\mathcal{R}_j, \mathcal{G}_j, \mathcal{B}_j\}_{j=1,\ldots,N}$, corresponding to the dominant colors (in RGB color code) of the N clusters. This step provides an initial ground truth for the region colors of the original image for comparative purposes with the potential solutions of our method throughout the iterations.

4.4 IFS code optimization

This is the most critical step of the method. It applies a hybrid approach combining genetic algorithms and particle swarm optimization to determine the optimal number of contractive functions and the IFS code (through PSO) and the color map of each contractive function (through GA).

4.4.1 Genetic algorithms

Genetic algorithms (GAs) are metaheuristic search procedures based on the Darwinian principles of natural evolution and natural selection. They become popular thanks to the seminal work carried out in the 1970 s by J. Holland [39]. Since then, they have been widely used in optimization problems, specially in combinatorial optimization [40].

The general structure of the genetic algorithm is shown in Table 1. Genetic algorithms are population-based methods that proceed iteratively by generations: at each generation g, a GA considers a population of individuals, where each particular individual, accounting for a potential solution, is represented using a genetic representation. In this work, we use a discrete representation where the individuals $\mathbf{D}_k$ are vectors of N_i components (as many as the number of contractive functions of the IFS) of the form: $\mathbf{D}_k = (\mathbf{D}_k^1, \mathbf{D}_k^2, \ldots, \mathbf{D}_k^{N_i})$ where each $\mathbf{D}_k^j = (R_k^j, G_k^j, B_k^j)$, and the R_k^j, G_k^j, B_k^j components, accounting for the red, green, and blue color channels, take integer values within the range $0 - 255$.

Generally the initial population is selected randomly, but some knowledge about the specific problem can also be embedded into the initial population with the aim at improving the convergence speed. Since in our case no prior knowledge about the problem is assumed, the individuals for the genetic algorithm are initialized with random values taken from an integer uniform distribution on the range $0 - 255$ for each color component.

A critical aspect in many GA applications is the size of this initial population, denoted onwards as N^{GA}. If it is too small, the algorithm could converge too quickly, while if it is too large the algorithm could waste valuable computational resources. The population size is often chosen to be constant, although GA with dynamic population size are also possible [42]. In this paper, we consider a fixed population of $N^{GA} = 100$ individuals.

Each one of the potential solutions must be evaluated by using a fitness function; the result of this evaluation is typically understood as a measure of the adaptation for each individual, which is, in turn, a measure of the goodness of the solution for the given optimization problem. For the color optimization subproblem addressed in this work with genetic algorithms, we consider the *normalized RGB color difference error*, $\mathcal{C}$, given by:

$$\mathcal{C}(\mathcal{I}_o, \mathcal{I}_r) = \frac{1}{3\,N_l}\left[\sum_{i=1}^{N_l}\left(\frac{|\mathcal{R}_o^i - \mathcal{R}_r^i| + |\mathcal{G}_o^i - \mathcal{G}_r^i| + |\mathcal{B}_o^i - \mathcal{B}_r^i|}{255}\right)\right] \tag{7}$$

where $(\mathcal{R}_o^i, \mathcal{G}_o^i, \mathcal{B}_o^i)$ and $(\mathcal{R}_r^i, \mathcal{G}_r^i, \mathcal{B}_r^i)$ are the RGB color channels of the contractive functions of the original and the reconstructed images, respectively, and N_l indicates the number of contractive functions of the optimal solution found by the method.

After the initialization step, the evolution process in genetic algorithms works iteratively: at each iteration new populations are obtained using a selection process, based on the Darwinian concept of individual adaptation. This selection process is repeated over the generations and the selected individuals form the new population. In this work, the selection operator is implemented as a biased roulette wheel with slots weighted proportionally to the individual fitness values. Then, some genetic operators (crossover and mutation) are applied onto the new population [40, 43]. The individuals exhibiting the best adaptation measure increase their chance of reproducing and generating new individuals through crossover and mutation. These genetic operators are driven by some parameters, as listed in Table 2.

The *crossover* operator generates two new individuals (*offsprings*) by combination of different parts from two

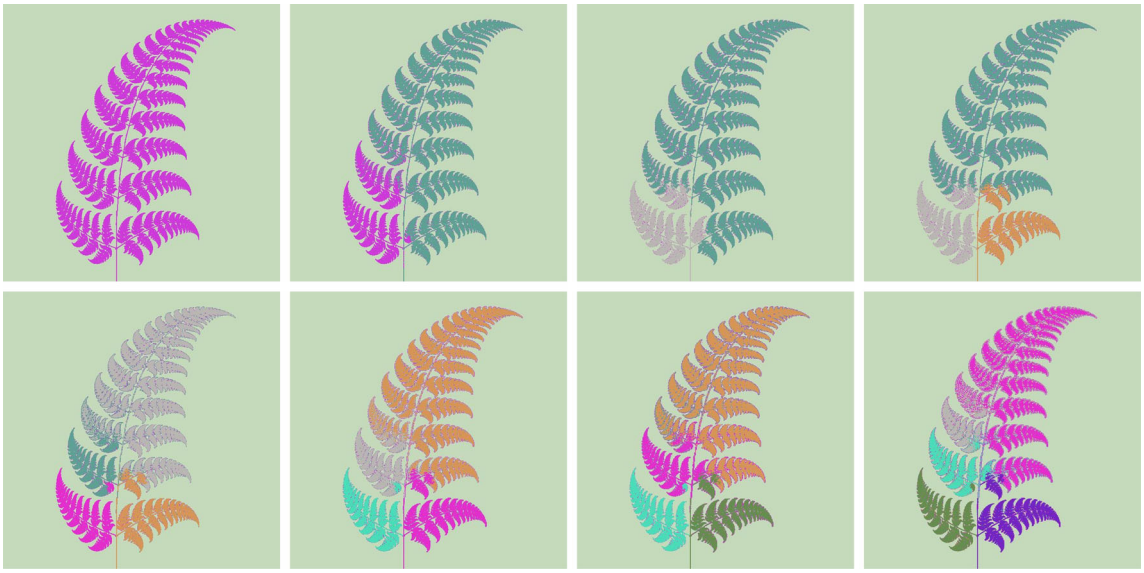


Fig. 3 k-means clustering results of the image in Fig. 2(left) for different number of clusters (left-right, top-bottom) ranging from 2 to 9 (including background)

Table 1 General structure of the genetic algorithm (modified from [41])

begin
  Let g=*0* be the generation counter
  Create and initialize a population, **Pop**$(0) = \{x_1(0), x_2(0), \ldots, x_p(0)\}$
  repeat
    Evaluate the fitness, $f(x_i(g))$, of each individual x_i of **Pop** *(g)*
    Select individuals from **Pop** *(g)*
    Apply crossover operator with probability p_c to produce offspring
    Apply mutation operator with probability p_m on offspring
    Set population of new generation **Pop**$(g+1)$
    Advance to the new generation $g = g + 1$
  until stopping condition is true
end

selected individuals of the population. In this work, we consider a two-point crossover operator that, with probability p_c, selects two random chromosomes of two individuals and then swaps the values of the parent chromosomes to produce the two new offsprings. A good performance of the GA requires the choice of a high crossover probability, in order to promote the diversity of the population. This will enhance the exploratory ability of the population, covering larger regions of the search space. We noticed that our crossover procedure can introduce strong perturbations to the individuals of the population even if they are close the convergence. To alleviate this effect, we store not only the new offsprings but also their parents, so that the population is temporarily increased. Then, the best individuals (as many as the GA population size, N^{GA}) are selected based on their individual fitness, while the rest of the population is rejected and, hence, not preserved for the next generation. This elitist strategy attenuates the effect of the crossover operator, while still promoting population diversity and the ability to explore regions of the search space far from the current best solutions.

Finally, a *mutation* operator is applied. It generates, with a given probability, p_m, a new individual by a small change in a single individual. The mutation operator guarantees that all the search space has a nonzero probability of being explored, by increasing further the diversity of the population. A good GA performance requires the choice of a low mutation probability (often taken inversely proportional to the population size). In our work, the mutation operator picks a random component of the individual, x_j and modifies it as $x_j \leftarrow x_j \pm m_s$, with m_s called the mutation strength, and where the positive or negative sign is randomly chosen with the same probability.

Table 2 Parameters of the genetic algorithm and values used in this paper

Symbol	Meaning	Used value
N^{GA}	Population size	100
p_c	Crossover probability	0.9
p_m	Mutation probability	0.1
m_s	Mutation strength	40

This procedure is repeated through successive generations until a termination condition is reached. Common terminating criteria are: (1) that a solution is found satisfying a lower threshold value; (2) that a fixed number of generations are reached; or (3) that successive iterations no longer produce better results. However, depending on the given problem, other criteria can be set up. In our case, the termination criterion is that no further improvement is attained after 10 consecutive generations.

4.4.2 Particle swarm optimization

Particle Swarm Optimization (PSO) is a highly popular bio-inspired metaheuristic technique for real-valued optimization problems. The original PSO algorithm was first proposed by Kennedy and Eberhart in 1995 [44]. Many other variants from this original procedure have been proposed [45]. The interested reader is referred to [46] for further details. See also [47] for a nice computational approach of the PSO and its variants.

In the PSO algorithm, an initial population of potential solutions (called *particles*) are distributed (uniformly in the general case) over the search space; then, they are provided with an initial velocity and the capacity to communicate the best positions to each other and adjust their own position and velocity based on these optimal positions. In this work, we consider a population size N^{PSO} of 100 particles. The particles $\mathcal{W}_i$, representing the IFS, follow a continuous representation given by vectors of N_i components (as many as the number of contractive functions of the IFS), of the form:

$$\mathcal{W}_i = (\phi_i^1, \phi_i^2, \ldots, \phi_i^{N_i}) \tag{8}$$

where each $\phi_i^j = (a_i^j, b_i^j, c_i^j, d_i^j, e_i^j, f_i^j)$, i.e., the IFS parameters of the corresponding contractive function ϕ_i^j. Without loss of generality, we can assume that the parameters $a_i^j, b_i^j, c_i^j, d_i^j, e_i^j, f_i^j$ take real values on the interval (0, 1]. However, since the functions ϕ_i^j must be contractive, the following constraints must hold:

$$\begin{cases} \left(a_i^j\right)^2 + \left(c_i^j\right)^2 < 1 \\ \left(b_i^j\right)^2 + \left(d_i^j\right)^2 < 1 \\ \left(a_i^j\right)^2 + \left(b_i^j\right)^2 + \left(c_i^j\right)^2 + \left(d_i^j\right)^2 < 1 + \left(a_i^j.d_i^j - b_i^j.c_i^j\right)^2 \end{cases} \tag{9}$$

These conditions are to be checked at every iteration step t.

Note that the length of each individual $\mathcal{W}_i$ is N_i, which can be different from one IFS to another. In other words, different to our previous conference paper, we do not consider an initial value for the number of contractive functions of the IFS. Instead, N_i is a dynamic variable that changes over the iterations according to the best fitness value at each iteration. In this way, the method is able to compute automatically the optimal number of contractive functions for any input fractal image. In this work, N_i is initialized randomly between 2 and 10, although larger values are also supported in our approach without further modification.

Similar to genetic algorithms, the dynamics of the population is considered along successive iterations. The evolution of the particle i is determined by two memory factors: the memory of their own best position, and knowledge of the global best. For the first main factor, we collect the coordinates P_i^b associated with the best solution (according to a given fitness function f) it has achieved so far along with the corresponding fitness value, $f(P_i^b)$. For the second main factor, we also store the position, P_g^b, and the best fitness value, $f(P_g^b)$, among all the particles in the population from the initial iteration. This global information is used to modify the position and velocity of each particle i according to the following evolution equations:

$$V_i(t+1) = w\,V_i(t) + \gamma_1 R_1\,[P_g^b(t) - P_i(t)] + \gamma_2 R_2\,[P_i^b(t) - P_i(t)] \quad (10)$$

$$P_i(t+1) = P_i(t) + V_i(t) \quad (11)$$

where $P_i(t)$ and $V_i(t)$ are the position and the velocity of particle i at time t, respectively, w is the *inertia weight* which defines how much the old velocity will affect the new one, and the coefficients γ_1 and γ_2 are constant values called *learning factors*, defining the level of affection of P_g^b and P_i^b. To be more precise, γ_1 is a weight accounting for the "social" or "global" component of the swarm, while γ_2 represents the "cognitive" or "personal" component, accounting for the memory of the individual particle i along the time. Two random numbers, R_1 and R_2, following a uniform distribution on the unit interval [0, 1], are included to diversify the search.

The PSO is used in this work to compute the optimal number of contractive functions and their IFS code. To this purpose, the fitness function computes the distance between the attractor of the reconstructed IFS and the input image. Since the input consists of 2D bitmap images given as a collection of pixels, the most natural choice would be the Hausdorff distance, given by Eq. (2). However, this metric is very computational expensive. Even worse, it may become unreliable for this problem, as it might identify as similar images that are different in terms of their geometry. For these reasons, other similarity functions have been proposed in the literature [25, 30, 32]. In this paper, we consider two of them, described in next paragraphs.

The first one is given by the Hamming distance, which has already been used in our previous conference paper in [7]. To compute this distance, we consider the binarized version of the original and the reconstructed images, denoted as $\mathcal{I}_o^B$ and $\mathcal{I}_r^B$, respectively, with the same size $p \times q$, and compute the *Hamming similarity error*, $\mathcal{S}_{\mathcal{H}}$, given by:

$$\mathcal{S}_{\mathcal{H}}(\mathcal{I}_o^B, \mathcal{I}_r^B) = e\,\frac{1}{p \times q}\sum_{x=1}^{p}\sum_{y=1}^{q}\left|\mathcal{I}_o^B(x,y) - \mathcal{I}_r^B(x,y)\right| \quad (12)$$

with $\mathcal{I}_j^B(x,y)$ indicating the binary value of the pixel (x, y) of $\mathcal{I}_j^B$. Note that $\mathcal{S}_{\mathcal{H}}(\mathcal{I}_o^B, \mathcal{I}_r^B)$ computes the number of mismatched pixels between both images. Therefore, values of $\mathcal{S}_{\mathcal{H}}(\mathcal{I}_o^B, \mathcal{I}_r^B)$ close to 0 mean that the images are very similar, indicating a very good geometrical reconstruction, while values close to 1 mean that both images are very different, a clear indication of poor reconstruction.

The second fitness function is the *intersection similarity function*, $\mathcal{S}_{\#_p}$, given by:

Table 3 Parameters of the particle swarm optimization method and values used in this paper

Symbol	Meaning	Used value
N^{PSO}	Population size	100
γ_1	Global factor	2.0
γ_2	Cognitive factor	2.0
w	Inertia weight	Linearly decreasing from 0.9

$$\mathcal{S}_{\#_p}(\mathcal{I}_o^B, \mathcal{I}_r^B) = \frac{\#_p(\mathcal{I}_o^B \cap \mathcal{I}_r^B)}{\#_p(\mathcal{I}_o^B \cup \mathcal{I}_r^B)} \quad (13)$$

where $\#_p(.)$ represents the number of active pixels of the image, that is, those displayed in black in the binarized version of the image; see, for instance, Fig. 2(right). In this case, values of this function close to 1 indicate very good matching, with the limit case for value 1, which corresponds to a perfect matching between the union and intersection of both images. The opposite applies for values approaching to 0.

The PSO also comes with some parameters, as shown in Table 3. The most critical parameters might be the social and cognitive factors. However, in our trials, we did not notice significant differences in the final results, only in the CPU times. We finally set them to 2.0, values that show a good behavior toward convergence, according to some theoretical studies. About the inertia weight, although it can be taken constant, some studies suggest that it is better to modify it dynamically over the time [48]. Initially, a relatively high value is generally considered, corresponding to a system with low viscosity so that the swarm performs extensive exploration. Then, the value is gradually decreased until a small value the system corresponding to a dissipative system where the swarm is better at homing into local optima. This is also the strategy taken in this paper: starting with a value $w = 0.9$, it is gradually decreased by 0.05 every 100 iterations.

This PSO algorithm is sketched in the pseudocode shown below. The procedure is repeated iteratively, and the solutions are evaluated at each iteration according to the similarity function given by Eq. (12) until a termination condition is reached. Once the method ends, the best particle of the swarm at the last iteration is taken as the best solution of the optimization problem.

Algorithm PSO Pseudocode version of the PSO algorithm (modified from [49]).

{Initialization}
  $t \leftarrow 0$ /* t: time variable */
  for $i = 1$ to N **do** /* N: size of the swarm */
    Initialize vectors V_i and P_i to random values
    $P_i^b \leftarrow P_i$
  end for
  $P_g^b \leftarrow best\{P_i^b\,; i = 1, \ldots, N\}$ /* initial global best */

{Main Loop}
  while (not termination condition) **do**
    {Evaluation Loop}
      for $i = 1$ to N **do**
        if $f(P_i)$ is better than $f(P_i^b)$ **then** /* f: fitness function */
          $P_i^b \leftarrow P_i$ /* particle's best position */
        end if
        if $f(P_i^b)$ is better than $f(P_g^b)$ **then**
          $P_g^b \leftarrow P_i^b$ /* swarm's best position */
        end if
      end for
    {Update Loop}
      for $i = 1$ to N **do**
        $V_i \leftarrow w.V_i + \gamma_1.R_1(0,1).(P_g^b - P_i) + \gamma_2.R_2(0,1).(P_i^b - P_i)$
        $P_i \leftarrow P_i + V_i$
      end for
      $t \leftarrow t + 1$
  end while

4.4.3 Hybridization of GA and PSO

Despite their remarkable features, genetic algorithms and particle swarm optimization also have some important limitations. For instance, PSO suffers from premature convergence (particles in PSO can get stuck in local minima or confined within a poor region of the search space). At its turn, genetic algorithms suffer from memory loss (whenever an individual in GA is not selected, its information is lost). Both are also affected by parameter tuning. In this context, the underlying idea of hybridization is to take advantage of their individual strengths to improve their performance while simultaneously overcoming its main limitations. In fact, hybrid evolutionary systems based on GA and PSO have been extensively used so far, with a number of papers showing that they outperform their individual components for several optimization problems (a brief discussion of several hybrid approaches can be found in [50] and references within).

In the particular case of this paper, GA and PSO are used to deal with the subproblems of dominant color optimization and IFS parameter optimization, respectively. Note that the color problem is discrete in nature, leading to a discrete optimization problem, where GA is a very suitable approach. On the contrary, the IFS parameters are real-valued, so their optimization is more suited for a continuous optimizer such as PSO. There is another important reason to consider hybrid schemes in this paper. The reconstruction of colored fractal images with IFS is affected by the fact that the computation of the IFS parameters and the color of each contractive function are strongly intertwined in a highly nonlinear way. Modifying the IFS parameters changes the contractive functions, which in turn modifies the attractor of the image and hence its colors. In order to match the new colors of the image again, new computations have to be carried out, which modify the IFS parameters, and so on.

In order to solve this problem, we consider a hybrid approach already used in the past to solve a different

problem in another field but with similar characteristics of nonlinear interplay between sets of free variables [50]. The method, called *IMCH GAPSO* (standing for *I*terative *M*utually *C*oupled *H*ybrid *GA-PSO*), is based on the idea that GA and PSO are mutually coupled, in the sense that the output of one system is used as the input of the other and vice versa. This coupling is then repeated iteratively until a termination criterion is attained. In particular, the GA is applied to optimize the color values of the contractive functions of the IFS, as explained in Sect. 4.4.1. Once a value for them is obtained, this input is injected into the PSO to determine the optimal IFS parameters with such colors, as described in Sect. 4.4.2. Once the PSO is executed, the fitness of the new particles along with the colors obtained from the GA is evaluated according to the global fitness function:

$$F(\mathcal{I}_0, \mathcal{I}_r) = \alpha \mathcal{S}_{\mathcal{H}}(\mathcal{I}_0, \mathcal{I}_r) + (1 - \alpha)\left[1 - \mathcal{C}(\mathcal{I}_0, \mathcal{I}_r)\right] \tag{14}$$

where α is a scalar factor taking values on the interval (0, 1). Equation (14) is a convex combination of the fitness functions associated with the two subproblems of this paper. In this sense, it extends the global fitness function in our previous conference paper [7], where we took $\alpha = 0.5$. This time, we carried out several simulations for different values of α varying from 0.1 to 0.9 with step size 0.1. From them, we found empirically that $\alpha = 0.7$ provides the best results in terms of accuracy and CPU times, so this is the value selected in this paper. The corresponding solutions are then ranked according to their fitness values and the process is started again with a new round of executions for the GA, then the PSO, and so on. This process is repeated for a given number of iterations, for which convergence has already been achieved. This number is set to 12,000 iterations in this paper.

4.5 Solution refinement through local search

Once the best solution is obtained through the IMCH GA-PSO method described in the previous section, it is combined with a local search procedure with the aim at improving its accuracy even further. In this paper, we consider two local search heuristics to intensify the search in the neighborhood of the global optima. The first procedure is the Luus–Jaakola local search procedure, which is a popular heuristic for nonlinear programming problems [51]. In this method, an initialization step is performed, where random uniform values are sampled for each component. These random values are added to the current values of the obtained solution to generate a new candidate solution, which replaces the current one in case of improvement of the fitness value. Otherwise, the sampling space is multiplicatively decreased by a factor, which can be assumed to be constant. This process is repeated iteratively. With each iteration, the neighborhood of the point decreases, and the search is increasingly limited to a smaller area until a termination criterion is reached.

The second local search procedure is called MADS (standing for *M*esh *Ad*aptive *S*earch algorithm). It is a direct search method originally proposed by Audet and Dennis in [52] for nonlinear optimization. A great advantage of this method is that it extends the family of search methods known as generalized pattern search by allowing local exploration in an asymptotically dense set of directions in the space of optimization variables. In other words, the local exploration of variables is no longer restricted to a finite number of directions, making MADS less restrictive for exploration of the search space than the classical pattern search methods. In this paper, we follow the implementation described in [53]. The reader is kindly referred there for further details.

The local search heuristics is run for 1,000 additional iterations during the local refinement step to further improve the quality of the solution. This scheme has shown to be very effective for the problem addressed in this paper, as it will be discussed in the next section.

5 Computational experiments and results

This section presents the main results obtained for the IFS image reconstruction problem of the *fern* fractal image in Fig. 2. Firstly, the graphical results are discussed. Then, the numerical results are presented and analyzed.

5.1 Graphical results

Figures 4, 5, 6, 7 and 8 show the evolution of the best solution of the population for the 13,000 iterations of our method with step-size 250, starting with a random initial image. Each picture in those images is comprised of two parts: on the left, the colored attractor of the best IFS is depicted, where the color of each contractive map is determined by the genetic algorithm and the geometry by the PSO; on the right, this colored attractor is combined with the target image (in black) superimposed on the attractor for easier and prompt visualization of the difference between both images. The images are also part of two *QuickTime* videos: *Video1.mov* (length: 110 s; size: 10.5 MB) and *Video2.mov* (length: 158 s; size: 13.9 MB), submitted as accompanying material of this paper.

The first image in Fig. 4(top-left) corresponds to one of the 100 random images generated for the initial population of the method. As clearly shown in this picture, the initial random image is extremely different to the input image. As

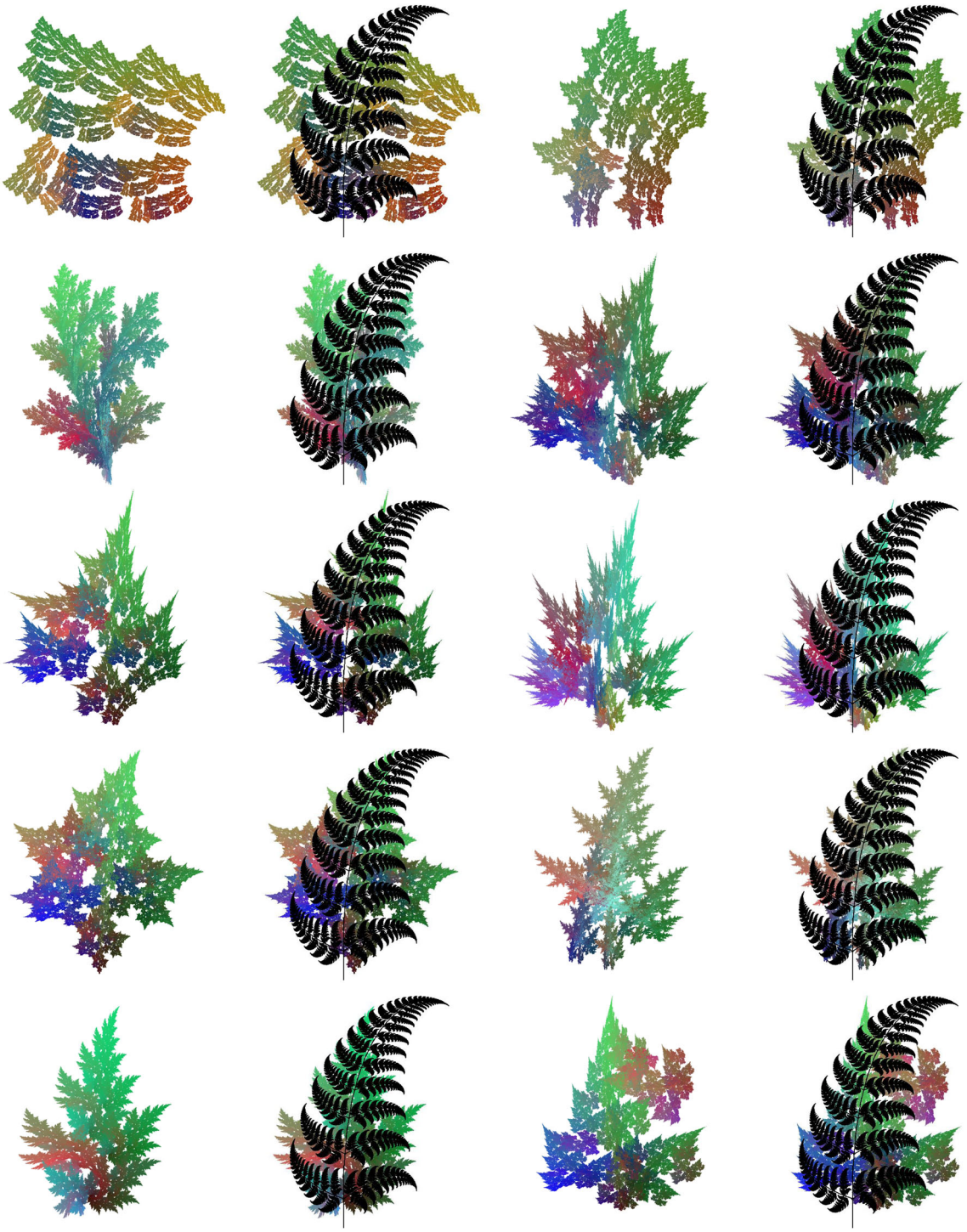


Fig. 4 (l-r, t-b) Evolution of the best solution for 0 to 2,250 iterations (step-size 250)

shown in the sequence of images in Figs. 4, 5, 6, 7 and 8, the application of our method makes the best solution to get closer to the target image over the iterations, until reaching a very good approximation of the input image at the last iterations. This convergence process is not visually obvious at the earlier iterations, as the strong nonlinear interplay between the color and the geometry can yield solutions that that might look worse than the previous ones from a visual point of view, either in terms of geometry or in terms of the intended color. On the other hand, we recall that the number of contractive maps of the IFS changes dynamically over the iterations. And since each contractive map is

Fig. 5 (l-r, t-b) Evolution of the best solution for 2,500 to 4,750 iterations (step-size 250)

associated with one color, this can make new colors appear during the iterative process. During our simulations, we noticed that the number of contractive functions of the partial best solutions typically tends to vary from 3 (the lowest value found in our simulations) to 10 (the largest one). For instance, the reader can see that a new blue color appears in the transition between the left and the right images of the second row of Fig. 4, indicating that a new contractive function has been added by the method. This new function disappears in the first row of Fig. 5, an indication that a contractive map has been removed from the IFS. New functions are automatically added or removed

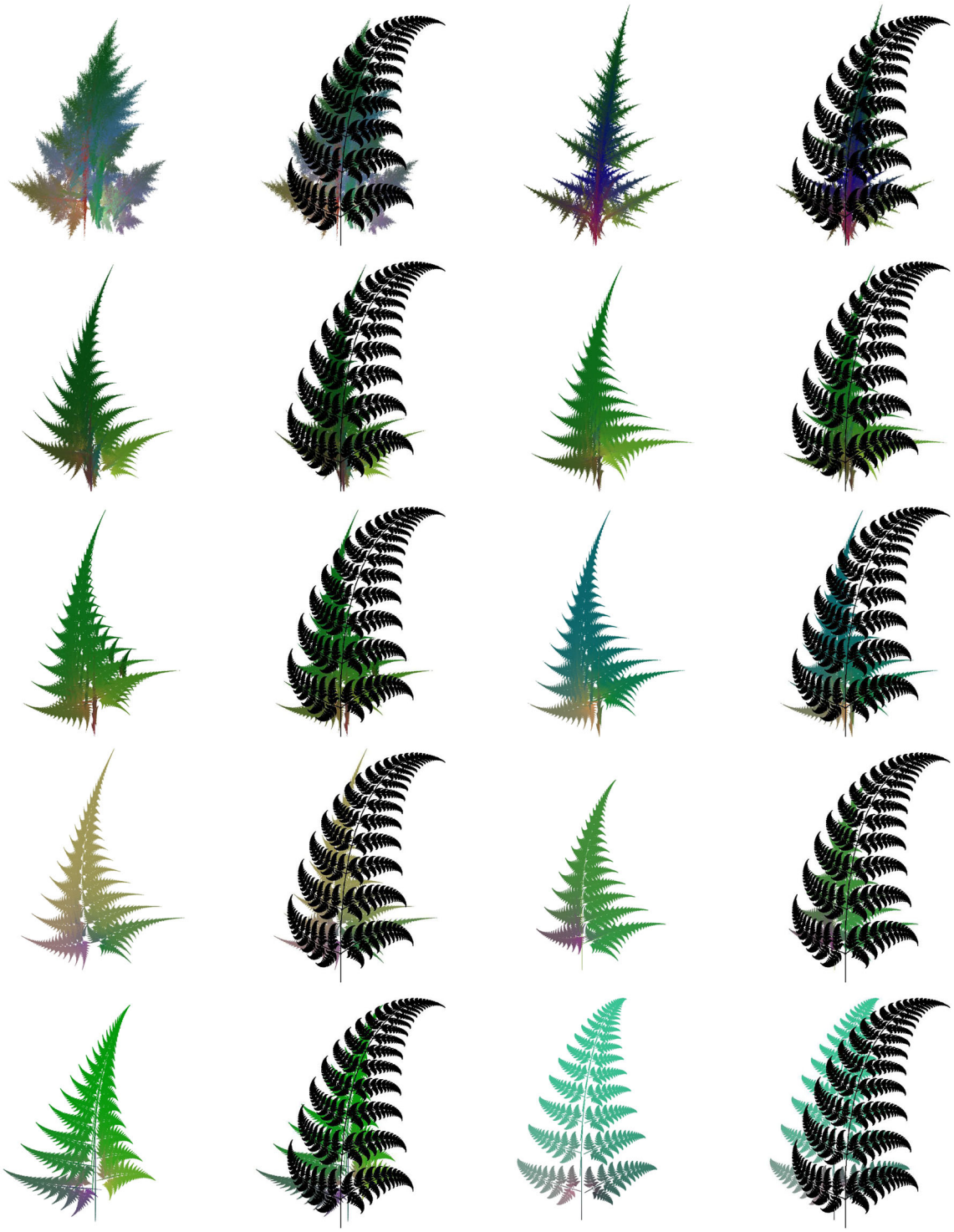


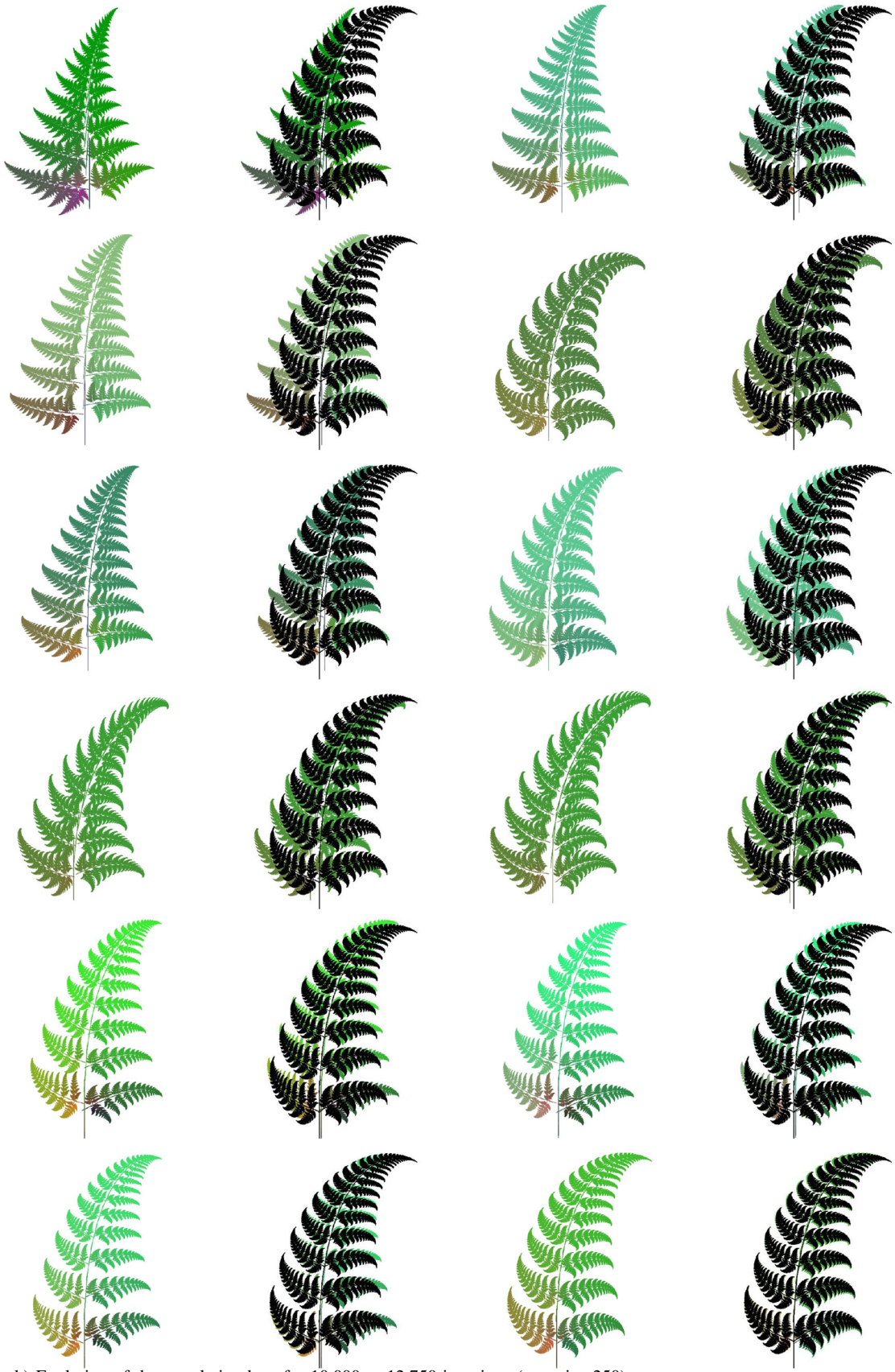
Fig. 6 (l-r, t-b) Evolution of the population best for 5,000 to 7,250 iterations (step-size 250)

by the method according to the fitness values all over the iterative process. Anytime a new contractive map is added, a new color is automatically assigned to the map by the genetic algorithm. That is the reason that explains why we can see different variations of the color palette throughout Figs. 4, 5, 6, 7 and 8.

In addition to our primary geometry-based fitness function, the Hamming similarity error, in this paper we also consider the intersection similarity function, as given

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

by Eq. (13). This fitness function requires to compute the intersection and the union of the original and the reconstructed images. Figures 9, 10, 11, 12 and 13 (left-right, top-bottom) show the evolution of the intersection (on the left) and the union (on the right) sets of the input and the reconstructed fractal images of the `fern` example for the 13,000 iterations with step-size 250. Note the remarkable difference between the intersection and the union sets at early stages of the method. This difference is visually decreasing over the iterations, and the global shape of the



◀**Fig. 8** (l-r, t-b) Evolution of the population best for 10,000 to 12,750 iterations (step-size 250)

target fractal image becomes more recognizable at later stages of the procedure. Two additional *QuickTime* videos: *Video3.mov* (length: 158 s; size: 8.7 MB) and *Video4.mov* (length: 158 s; size: 7.6 MB), showing, respectively, the evolution of the intersection and the union sets for our

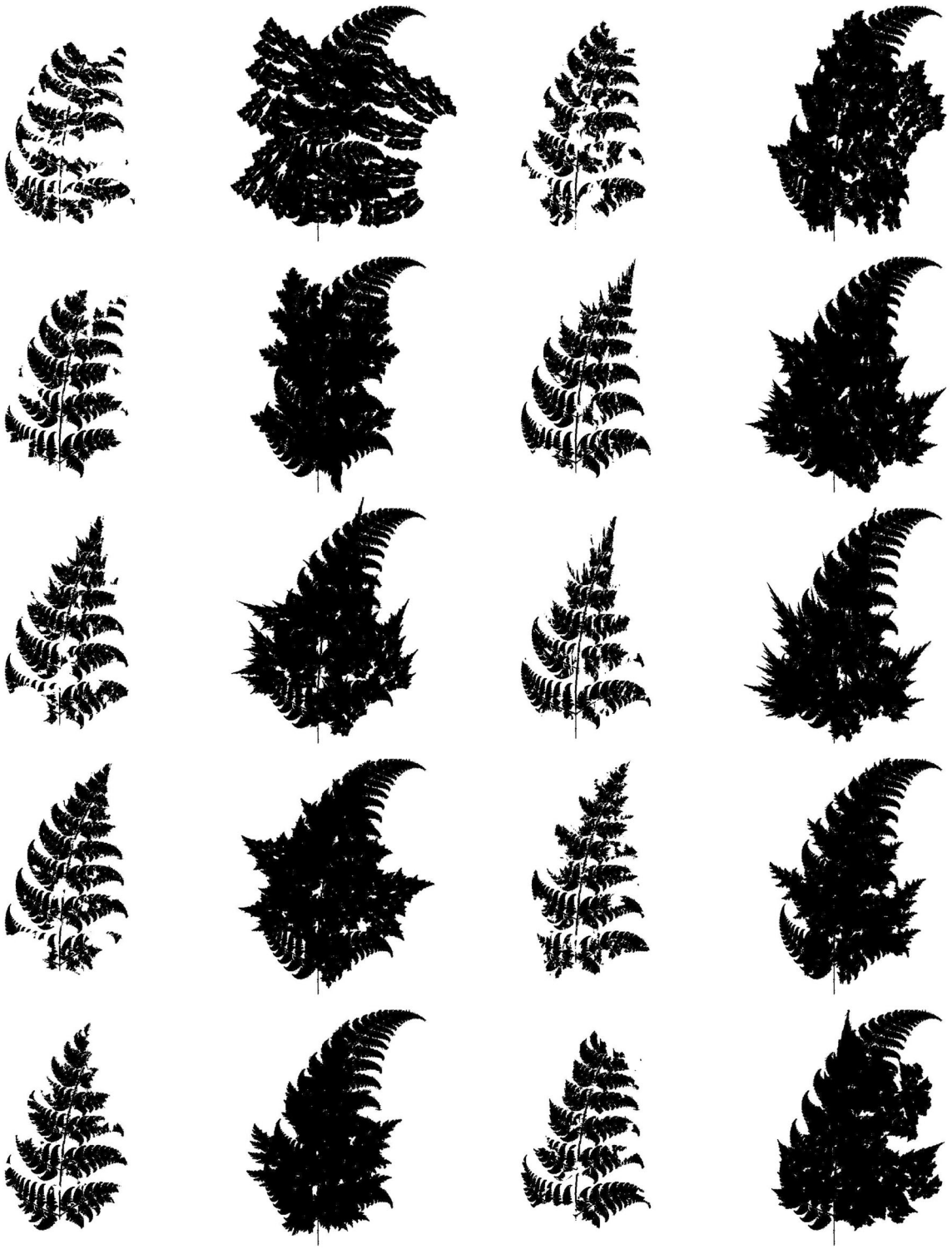


Fig. 9 (l-r, t-b) Intersection (left) and union (right) sets of the input and approximating fractal images of the `bush` example for 0 to 2,250 iterations (step-size 250)

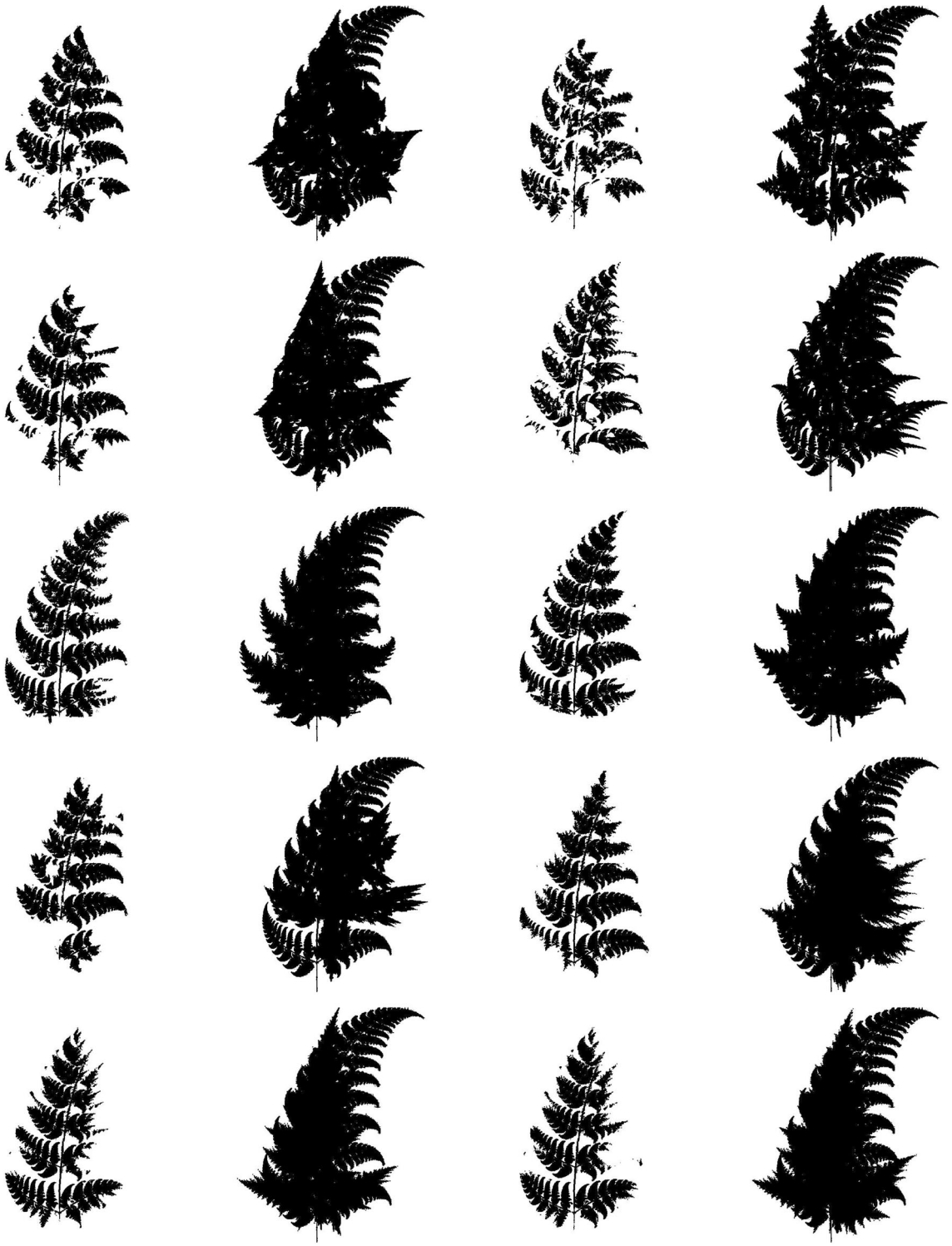


Fig. 10 (l-r, t-b) Intersection (left) and union (right) sets of the input and approximating fractal images of the `bush` example for 2,500 to 4,750 iterations (step-size 250)

example, are also submitted as accompanying material of the paper.

Figure 14 summarizes the graphical results of the global best of our method for the *fern* example in this paper. The top row shows the reconstructed colored image (on the left). A simple visual comparison with the original image in Fig. 2(left) shows that the proposed method performs very well, since the final reconstructed image is very

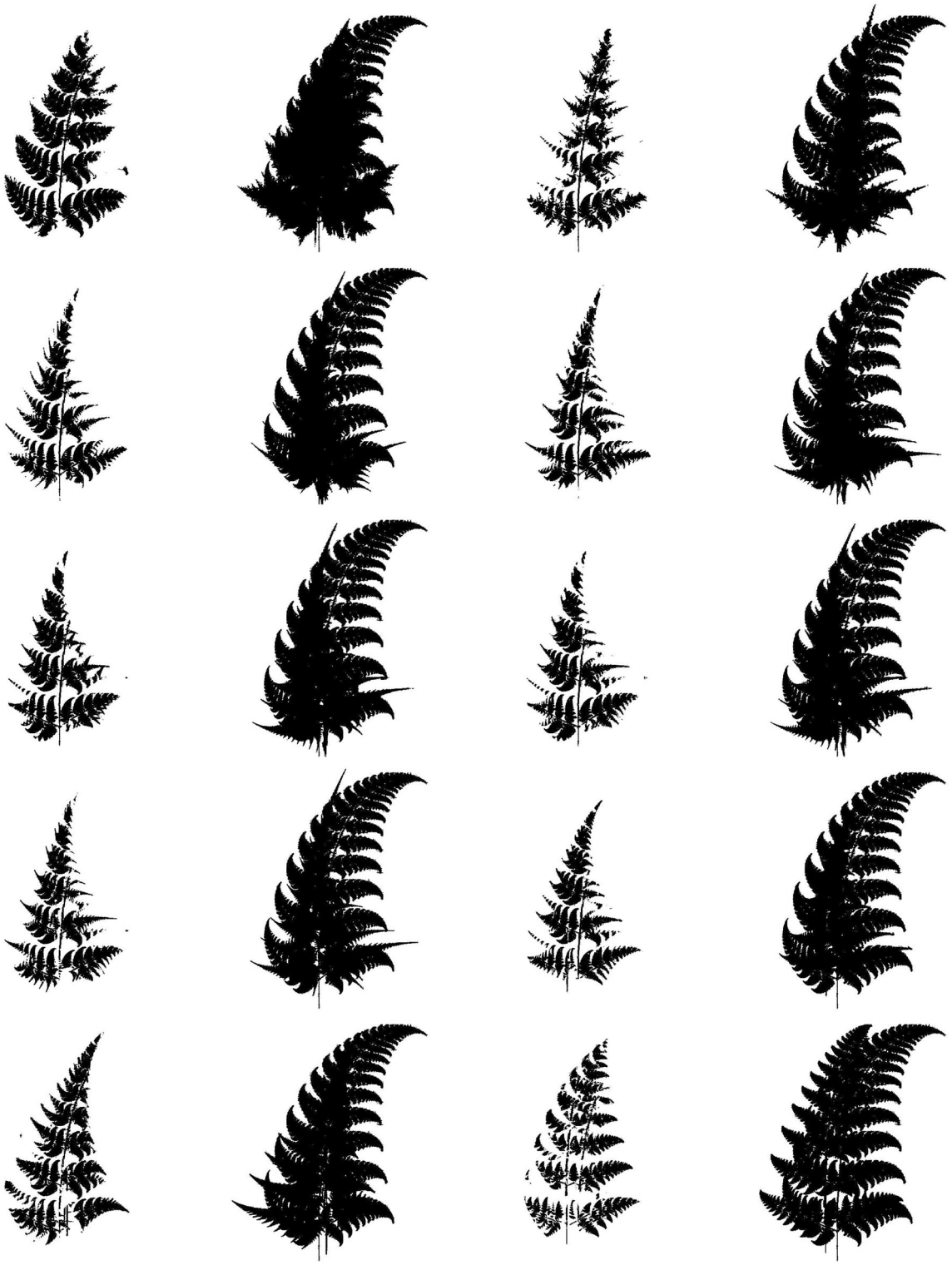


Fig. 11 (l-r, t-b) Intersection (left) and union (right) sets of the input and approximating fractal images of the `bush` example for 5,000 to 7,250 iterations (step-size 250)

similar visually to the target one. The image in Fig. 14(top-right) shows the superposition of the original image (in black) and the reconstructed image (in full color behind) for easier visual comparison. The fact that the colors of the reconstructed image in the background can barely be seen (as they are hidden by the original image in black in the

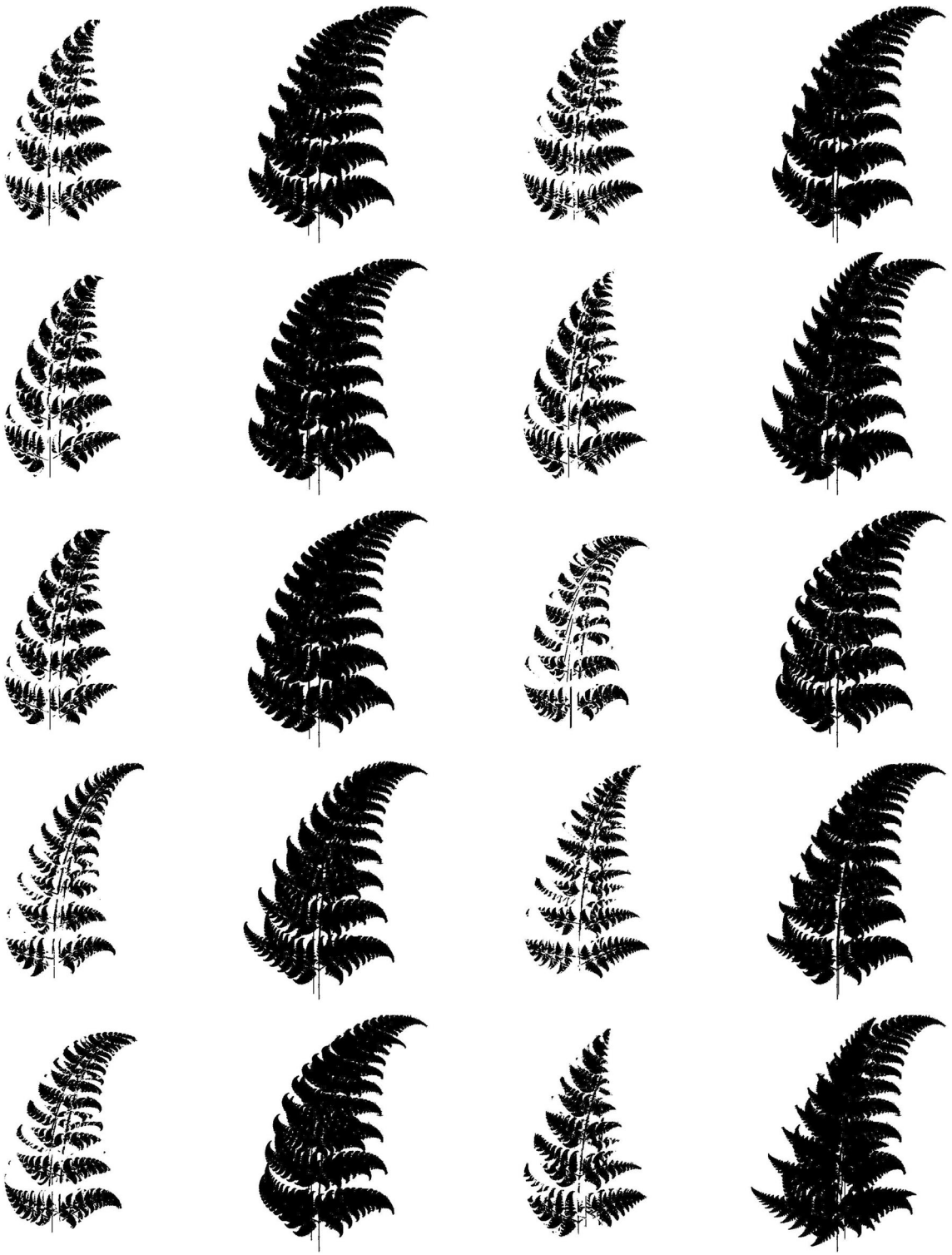
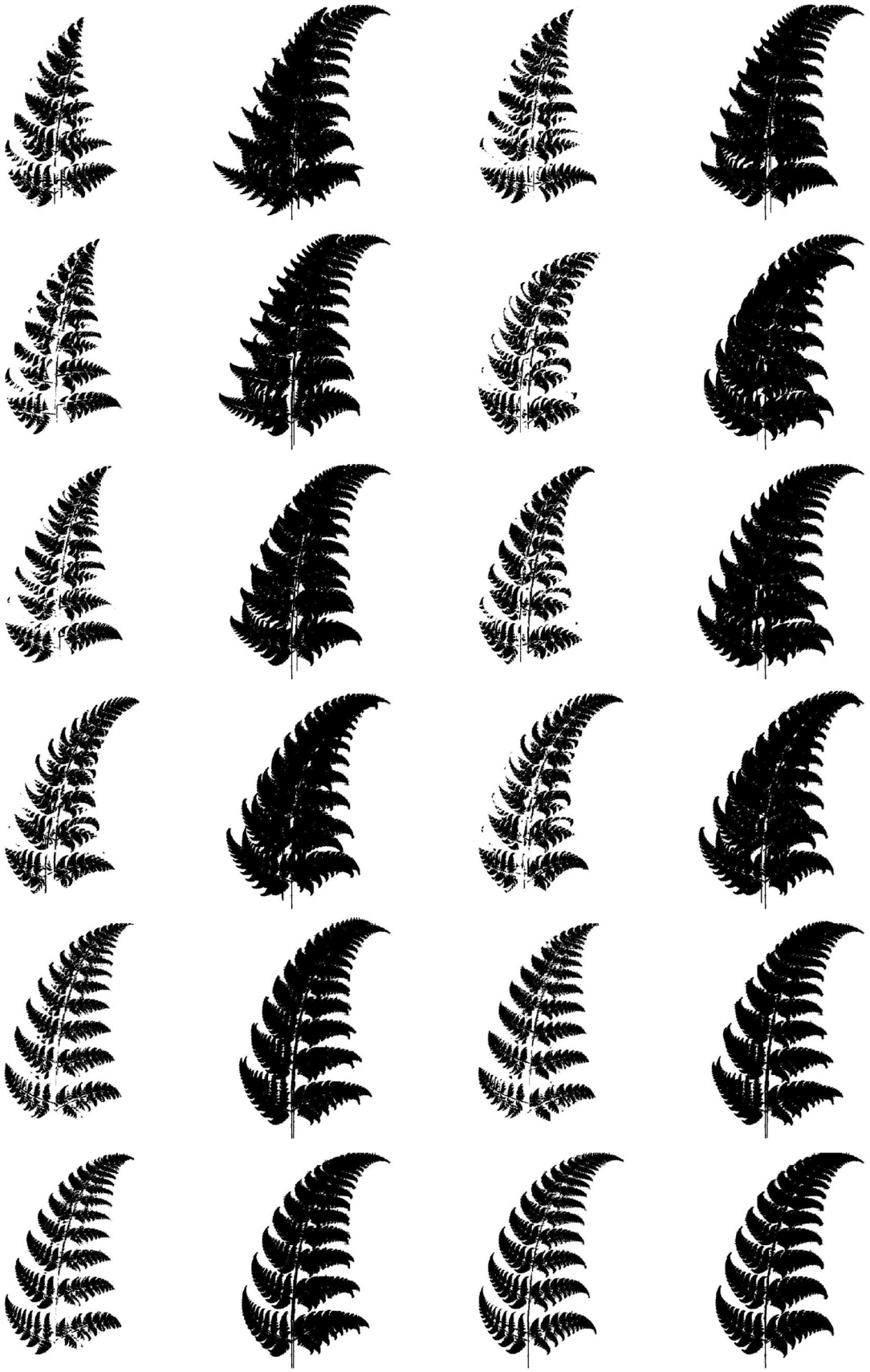


Fig. 12 (l-r, t-b) Intersection (left) and union (right) sets of the input and the approximating fractal images of the `bush` example for 7,500 to 9,750 iterations (step-size 250)

foreground) is a clear indication of the high accuracy of the geometric reconstruction. As a further evidence, the bottom row of Fig. 14 shows the intersection (left) and union (right) of the original and the reconstructed fractal images. Note that both images look quite similar, albeit not identical. The ground truth image is obviously in-between. These graphical results show that the method is able to



◀**Fig. 13** (l-r, t-b) Intersection (left) and union (right) sets of the input and the approximating fractal images of the `bush` example for 10,000 to 12,750 iterations (step-size 250)

capture successfully all major features of the input image, even although it has a complicated and irregular shape.

Figure 15 summarizes the graphical results regarding the reconstruction of color with our method. The figure shows the histogram of the original fractal image (left) and the reconstructed image (right), with the three color channels displayed individually (top) and combined with transparent colors for better visualization (bottom). Once again, the visual results show that the method performs very well: both histograms look very similar, although the matching is not perfect. The difference is more noticeable for the blue color channel, corresponding to the bottom rightmost areas of the fractal image, where the geometric matching is less accurate. Yet, in our opinion the results are quite impressive. To the best of our knowledge, there is no method reported in the literature even attempting to address this problem, much less getting this high accuracy results.

5.2 Numerical results

The good graphical results described in previous section have been confirmed by the numerical results. Table 4 summarizes the main results obtained for the example in this paper. It shows the results obtained for the global best solution for the 13,000 iterations with step-size 1,000 (in rows). The following data are shown (in columns):

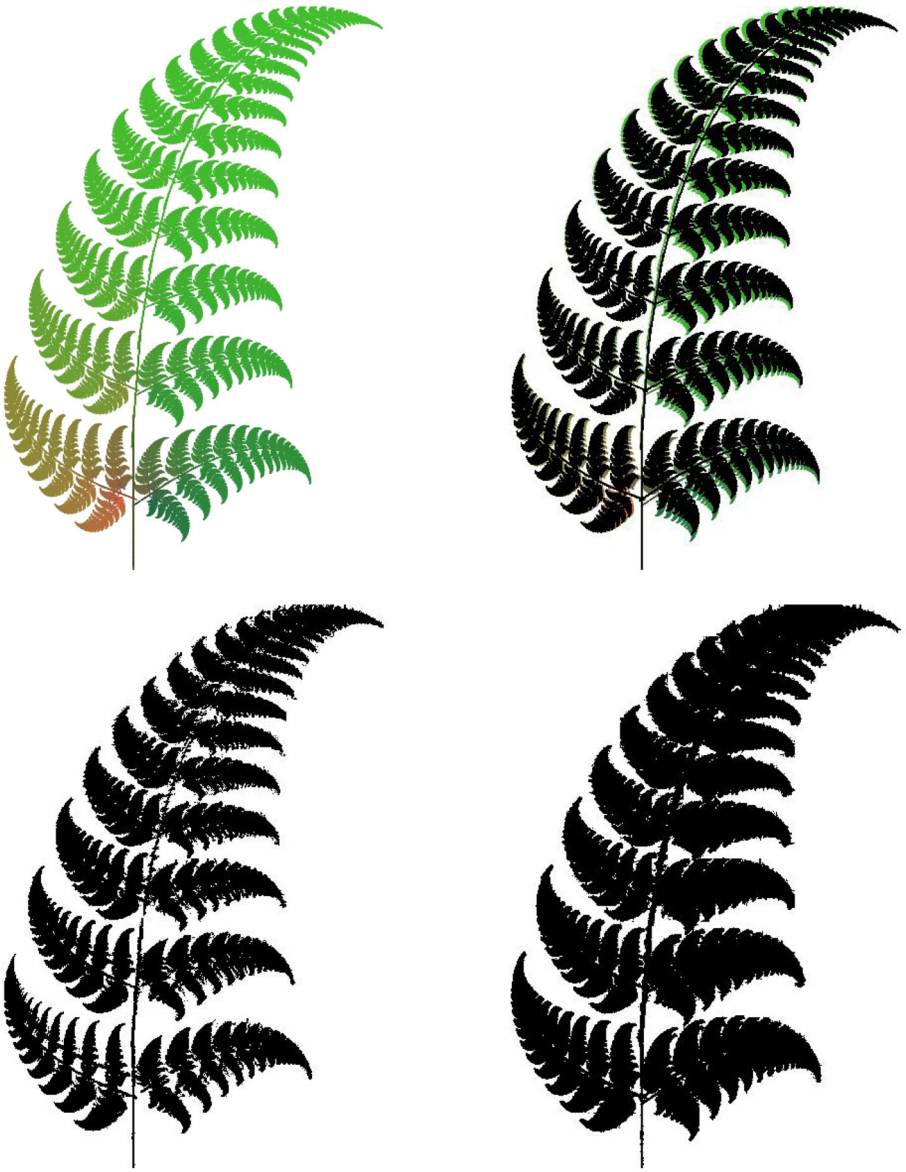


Fig. 14 Summary of the graphical results: (top-left) reconstructed colored image (compare it with the original one in Fig. 2(left)); (top-right) superposition of the original (in black) and the reconstructed (in color) images for visual comparison; (bottom) intersection (left) and union (right) of the original and the reconstructed fractal images

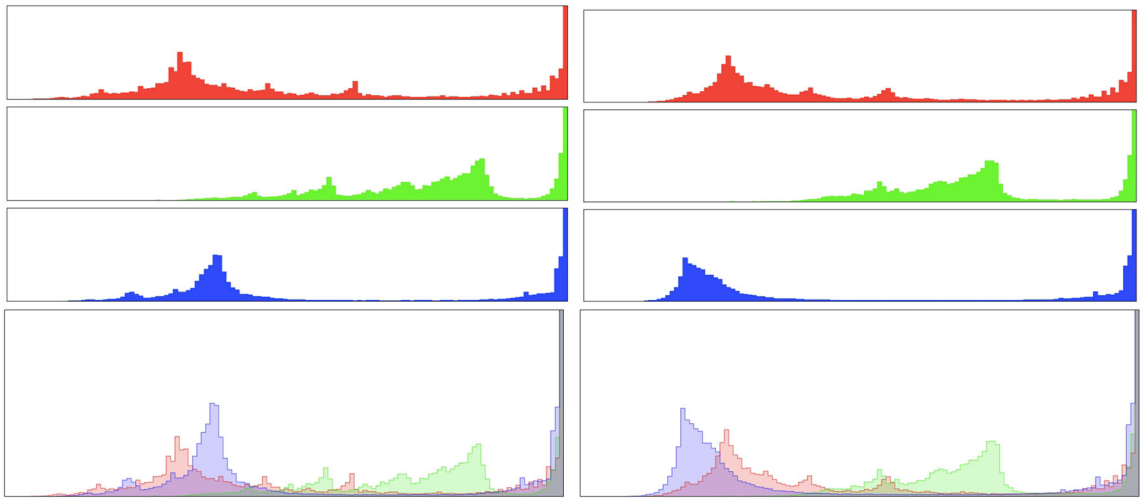


Fig. 15 Histogram of the original fractal image (left) and the reconstructed fractal image (right), with the three color channels displayed individually (top) and combined with transparent colors (bottom)

- Number of iterations, *iter*;
- Number of contractive maps, *N*, of the best solution at generation *iter*;
- Number of mismatched pixels between the reconstructed and the original images;
- Ratio of mismatches with respect to the active pixels of the image;
- Number of pixels in the intersection of the reconstructed and the original images;
- Number of pixels in the union of the reconstructed and the original images;
- Ratio between the number of pixels in the intersection and the union;
- Ratio between the number of pixels in the intersection and the number of active pixels of the image.

The most important conclusion from Table 4 is the excellent behavior of the method in terms of the geometric accuracy and its predictive capacity. Taking a look at the last row of the table, which displays the final results of the method, we can see that the method is able to recover the optimal number of contractive functions of the input image, which is $N = 4$, even although this datum is never used in our method. On the other hand, there are only 18,445 different pixels between the original and the reconstructed images, a ratio of only 22.98% of the total pixels of the image. Also, we remark that the number of pixels in the intersection and in the union is very close to the actual number of pixels of the target image, which is obviously between them. The disparity of these values is less than 10,000 pixels, a remarkable error of less than the 13%. Finally, the percentage of pixels in the intersection is

Table 4 Numerical results of our method for the *fern* fractal example

iter	*N*	≠pix	≠pix (ratio)	pix (int.)	pix (union)	int./union (ratio)	int./input (ratio)
0	5	119,509	1.4893	46,686	166,195	0.2809	0.5818
1,000	6	72,039	0.8977	53,033	125,073	0.4240	0.6609
2,000	6	57,947	0.7221	54,429	122,376	0.4843	0.6783
3,000	5	65,708	0.8188	43,746	109,454	0.3996	0.5451
4,000	7	66,198	0.8251	45,405	111,602	0.4168	0.5759
5,000	7	55,422	0.6907	55,571	110,993	0.5006	0.6925
6,000	5	64,998	0.8100	33,175	98,173	0.3379	0.4134
7,000	4	58,494	0.7289	39,796	98,290	0.4048	0.4959
8,000	4	67,990	0.8473	44,844	112,834	0.3974	0.5588
9,000	4	58,233	0.7257	39,609	97,842	0.4048	0.4936
10,000	4	60,547	0.7545	46,869	107,416	0.4363	0.5841
11,000	4	52,325	0.6521	49,585	101,910	0.4865	0.6179
12,000	4	40,335	0.5026	57,000	97,335	0.5856	0.7103
13,000	4	18,445	0.2298	70,164	88,609	0.7918	0.8744

larger than 87%, a clear indication of an excellent matching. We are unaware of any paper reporting numerical results with this high level of accuracy, even when considering methods focused exclusively on binary images without any color.

Table 4 provides other interesting information about the previous graphical results. The second column shows that the method is able to determine automatically the optimal number of contractive maps over the iterations. Note also that the value of N in the table changes from 4 to 7 over the generations. In addition, the values of N change even further over the population, taking values from 3 to 10, but these values do not appear in the table as they correspond to suboptimal solutions, while the table only shows the best solution for each iteration value. Also, it is interesting to remark that due to the interplay between the geometry and the color, it may happen that changing the number of contractive functions does not necessarily lead to an improvement of the geometric matching between the original and the reconstructed images. For instance, the numerical results are better for 2,000 iterations than for 3,000 or 4,000. However, this tells us only a part of the story, because the higher matching for 2,000 iterations is obtained by using contractive functions covering large areas, with a total of 122, 376 pixels for the union and only 54, 429 for the intersection. It is obvious that if we approximate the original image with other covering all its area in excess, we recover most of the pixels of the target image, but at the cost of adding many wrong pixels in excess to the image. This is exactly what happens here. These situations are allowed in our method in order to prevent the method from getting stuck in local optima and to increase the exploratory ability of the method, searching for more promising areas of the search space.

5.3 Computational issues

The computations in this work have been performed on a personal computer with 5.2 GHz Intel Core i9-11900 processor with 32 GB of RAM and graphical card Nvidia Quadro P620 2 Gb GDDR5X. The source code has been implemented by the authors in the native programming language of the popular scientific program *MATLAB version 2021b* and using the numerical libraries for fractals in [54, 55]. The clustering procedure has been run on *Mathematica* using source code adapted by the authors from public libraries for image processing and computer vision. The videos have been generated by the authors using the *iLife* suite by Apple. Regarding the time complexity of the method, taken into account our choices for the genetic operators of selection (roulette wheel), two-point crossover and one-point mutation, the genetic algorithm complexity is $O(g.N^{GA}.N_i)$, with g the number of generations. For the PSO, using our constant values for the social and cognitive factors and a linear variation of the inertia weight, the complexity is $O(T.N^{GA}.N_i)$ with T the number of iterations.

6 Conclusions and future work

This paper is an extension of a previous conference paper in [7] to address the problem of fractal image reconstruction for colored images through IFS in a fully automatic way. Given a colored fractal image, the goal is to obtain the IFS parameters and the color map of each contractive function automatically so that the input image can be accurately replicated not only in terms of its geometry but also its color, without any knowledge about the input fractal image beyond the input bitmap image. This work presents a method to address this issue based on four steps: image clustering, color determination, IFS code optimization, and local search refinement. The optimization task is solved through a hybrid method combining GA and PSO to compute the color of the contractive functions, and the optimal number of contractive functions and their IFS parameters, respectively. The GA and PSO algorithms are mutually coupled so that the input of each method is injected as input for the other, in an iterative fashion until convergence is achieved.

This new method has been applied to a popular yet challenging example of fractal image. The graphical and numerical results clearly show that the method performs very well, leading to excellent reconstruction results for both the geometry and the color. To the best of our knowledge, no other method has ever attempted to solve automatically the fractal image reconstruction problem with IFS for colored images, a clear evidence of the originality and contribution of this work. In this context, the present work opens the door for future developments in this field.

This work has also some limitations. The most noticeable is that only one example of fractal image is presented, raising a reasonable open question about the generality of this work. We have performed some trials on other examples and found that the method is also able to provide very good results, which could lead to a positive answer to this generality question. Unfortunately, we have had no time to finish all our simulations for the submission deadline of this special issue, and hence, these other examples are not included here. Of course, it would be great to include other images in the benchmark, but getting the results for this example has been an intensive work, requiring several days in simulations and data analysis. And this leads to what is, in our opinion, the most critical aspect of this method: the computational times. The method, as exposed above, provides a high degree of

accuracy at the expense of very large computational times. A typical execution of the method as described for the *fern* example takes tens of hours for a single run on a powerful personal computer. Given the stochastic nature of the metaheuristic techniques, several executions must be carried out in order to suppress spurious behaviors and obtain reliable results, which means that, unless you have a computer farm at your disposal (not certainly the authors' case), you must be ready for weeks of extensive computations. Clearly, the computational speed is still a challenge for this method. However, our focus in this work is not on computational speed, but on accuracy. Since no method has been reported so far in the literature to solve this problem, we wanted to confirm the feasibility of our approach to yield accurate results regardless of the computational effort, so we accepted the challenge and are extremely satisfied with the (arguably) impressive results. Given this pioneering nature, we have prioritized its precision over any other factor of the problem. But this large CPU time must be taken into account as a limiting factor of the method.

Once we show that the method performs well, our next focal point goes to the limitations of the method and how they can be improved. We are working now on ideas and new procedures to alleviate the computational load and decrease the computational times. One possibility could be to identify more efficient clustering methods for the color determination subproblem. Other line of research might be to propose more efficient fitness functions for this problem. These ideas will be part of our plans for future work in the field.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00521-023-08954-7.

Acknowledgements A. Gálvez and A. Iglesias thank the financial support from the projects PDE-GIR (Marie Sklodowska-Curie grant agreement No 778035) of the European Union's Horizon 2020 research & innovation program, and #PID2021-127073OB-I00 from the Agencia Estatal de Investigación, Spanish Ministry of Science and Innovation (Computer Science National Program) and European Funds EFRD (AEI/FEDER, UE). I.Fister and I. Fister Jr. thank the financial support from the Slovenian Research Agency (Grants No. P2-0041 and No. P2-0057, respectively).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability statement The datasets generated and/or analyzed during the current study will be available upon the acceptance of the paper from the corresponding author on reasonable request and with permission of all individuals and institutions involved in this study.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this article. Any commercial identity mentioned in this paper is cited solely for scientific purposes.

References

1. Falconer K (2003) Fractal geometry: mathematical foundations and applications, 2nd edn. Wiley, Chichester, England
2. Peitgen HO, Jurgens H, Saupe D (1993) Chaos and fractals. New Frontiers of science. Springer, New York
3. Barnsley MF (1993) Fractals everywhere, 2nd edn. Academic Press, San Diego
4. Hutchinson JE (1981) Fractals and self similarity. Indiana Univ Math J 30(5):713–747
5. Barnsley MF, Hurd LP (1993) Fractal image compression. AK Peters/CRC Press, London
6. Fisher Y (ed) (1995) Fractal image compression: theory and applications. Springer, Berlin
7. Gálvez A, Fister I, Deb S, Fister I, Fister I, Iglesias A (2021) Cuckoo search algorithm and K-means for IFS reconstruction of fractal colored images. In: Proceedings 8th international conference on soft computing & machine intelligence, ISCMI 2021, IEEE Computer Society Press, pp. 91–95
8. Barnsley MF, Demko S (1985) Iterated function systems and the global construction of fractals. Proc R Soc London A399:243–275
9. Barnsley MF, Ervin V, Hardin D, Lancaster J (1986) Solution of an inverse problem for fractal and other sets. Proc Natl Acad Sci 83:1975–1977
10. Gutiérrez JM, Iglesias A, Rodríguez MA (1996) A multifractal analysis of IFSP invariant measures with application to fractal image generation. Fractals 4(1):17–27
11. Barnsley MF, Sloan AD (1988) A better way to compress images. BYTE Mag
12. Jacquin AE (1992) Image coding based on a fractal theory of iterated contractive image transformations. IEEE Trans Image Process 1(1):18–30
13. Saupe D, Hamzaoui R (1994) A review of the fractal image compression literature. Comput Graph 28(4):268–276
14. Abiko T, Kawamata M (1999) IFS coding of non-homogeneous fractal images using Gröbner basis. In: Proceedings of the IEEE international conference on image processing, pp. 25–29
15. Berkner K (1997) A wavelet-based solution to the inverse problem for fractal interpolation functions. In: L Véhel et al. (eds.) Fractals in engineering'97. Springer Verlag
16. Wadstromer N (1997) An approach to the inverse IFS problem using the Kantorovich metric. Fractals 5(1):89–99
17. Vyrscay ER (1991) Moment and collage methods for the inverse problem of fractal construction with iterated function systems. In: Peitgen HO et al. (eds.) Fractals in the fundamental and applied sciences. Elsevier
18. Goentzel B (1994) Fractal image compression with the genetic algorithm. Complex Int 1:111–126

19. Zheng Y, Liu GR, Niu XX (2006) An improved fractal image compression approach by using iterated function system and genetic algorithm. Comput Math Appl 51:1727–1740
20. Lutton E,Véhel JL, Cretin G, Glevarec P, Roll C (1995) Mixed IFS - resolution of the inverse problem using genetic programming. INRIA Rapport 2631
21. Nettleton DJ, Garigliano R (1994) Evolutionary algorithms and a fractal inverse problem. Biosystems 33:221–231
22. Wu MS, Teng WC, Jeng JH, Hsieh JG (2006) Spatial correlation genetic algorithm for fractal image compression. Chaos Solitons Fractals 28(2):497–510
23. Wu MS, Jeng JH, Hsieh JG (2007) Schema genetic algorithm for fractal image compression. Eng Appl Artif Intell 20:531–538
24. Yuan WX, Ping LF, Guo WS (2009) Fractal image compression based on spatial correlation and hybrid genetic algorithm. J Vis Commun Image R 20:505–510
25. Dasgupta D, Hernandez G, Niño F (2000) An evolutionary algorithm for fractal coding of binary images. IEEE Trans Evolut Comput 4(2):172–181
26. Evans AK, Turner MJ (1998) Specialisation of evolutionary algorithms and data structures for the IFS inverse problem. In: M.J. Turner (Ed.). In: Proceedings of the second IMA conference on image processing: mathematical methods, algorithms, and applications
27. Shonkwiler R, Mendivil F, Deliu A (1991) Genetic algorithms for the 1-D fractal inverse problem. In: Proceedings of the fourth international conference on genetic algorithms, Morgan Kaufmann, pp. 495–501
28. Muruganandham A, Wahida RSD (2010) Adaptive fractal image compression using PSO. Proc Comput Sci 1:338–344
29. Tseng CC, Hsieh JG, Jeng JH (2008) Fractal image compression using visual-based particle swarm optimization. Image Vis Comput 26:1154–1162
30. Gálvez A, Iglesias A (2019) Modified bat algorithm with local search for fractal image compression of bitmap images. In: Proceedings of international conference on cyberworlds, CW 2019, IEEE Computer Society Press, Los Alamitos, CA, pp. 199–206
31. Gálvez A, Iglesias A (2019) Modified memetic self-adaptive firefly algorithm for 2D fractal image reconstruction. In: Proceedings of IEEE 42nd annual computer software & applications conference, IEEE-COMPSAC 2019. Tokyo (Japan), IEEE, pp. 165–170
32. Gálvez A, Iglesias A, Díaz JA, Fister I, López J, Fister I (2021) Modified OFS-RDS bat algorithm for IFS encoding of bitmap fractal binary images. Adv Eng Inf 47:101222
33. Gálvez A, Fister I, Fister I, Iglesias A (2021) Image reconstruction of colored bitmap fractal images through bat algorithm and color-based image clustering. In: Proceedings 16th international conference on soft computing models in industrial and environmental applications, SOCO 2021, Advances in Intelligent Systems and Computing, vol 1401, pp. 222–232
34. Elton JH (1987) An ergodic theorem for iterated maps. Ergodic Theory Dyn Syst 7:481–488
35. Gutiérrez JM, Iglesias A, Rodríguez MA, Rodríguez VJ (1997) Generating and rendering fractal images. Math J 7(1):6–13
36. Graf S (1992) Barnsley's scheme for the fractal encoding of images. J Complex 8:72–78
37. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66
38. Liao PS, Chen TS, Chung PC (2001) A fast algorithm for multilevel thresholding. J Inf Sci Eng 17(5):713–727
39. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
40. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley
41. Gálvez A, Iglesias A (2012) Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction. Inf Sci 182(1):56–76
42. Goldberg DE (1985) Optimal initial population size for binary-coded genetic algorithms, TCGA Report No.85001. University of Alabama
43. Mitchell M (1998) An introduction to genetic algorithms (complex adaptive systems). MIT Press, London
44. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: IEEE international conference on neural networks, Perth, Australia pp. 1942-1948
45. Eberhart RC, Shi Y (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 congress on evolutionary computation, pp. 81-86
46. Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann Publishers, San Francisco
47. Engelbretch AP (2005) Fundamentals of computational swarm intelligence. Wiley, Chichester, England
48. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization: an overview. Swarm intelligence 1:33–57
49. Gálvez A, Iglesias A (2012) Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. Inf Sci 192(1):174–192
50. Gálvez A, Iglesias A (2013) A new iterative mutually coupled hybrid GASO approach for curve fitting in manufacturing. Appl Soft Comput 13(3):1491–1504
51. Luus R, Jaakola THI (1973) Optimization by direct search and systematic reduction of the size of search region. Am Inst Chem Eng J (AIChE) 19(4):760–766
52. Audet C, Dennis JE Jr (2006) Mesh adaptive direct search algorithms for constrained optimization. SIAM J Optim 17(1):188–217
53. Iglesias A, Gálvez A, Avila A (2013) Hybridizing mesh adaptive search algorithm and artificial immune systems for discrete rational Bzier curve approximation. Vis Comput 32(3):393–402
54. Gálvez A (2009) IFS Matlab generator: a computer tool for displaying IFS fractals. In: Proceedings ICCSA'2009, EEE CS Press, Los Alamitos, CA, pp. 132–142
55. Gálvez A, Kitahara K, Kaneko M (2014) IFSGen4LaTeX: interactive graphical user interface for generation and visualization of iterated function systems in LaTeX. Lect Notes Comput Sci 8592:554–561