Improved Cuckoo Search with Luus-Jakoola Heuristics for the IFS Inverse Problem of Binary Self-Similar Fractal Images

Akemi Gálvez[1,2] and Andrés Iglesias[1,2(✉)]

[1] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Narashino Campus, Funabashi 274-8510, Japan
[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avenida de los Castros s/n, 39005 Santander, Spain
iglesias@unican.es
http://personales.unican.es/iglesias

Abstract. This paper addresses the following problem: how to reconstruct a given binary self-similar fractal image through iterated functions systems. This means to obtain an iterated function system (IFS) whose attractor is a good approximation of the input image. This problem is known to be a very difficult multivariate nonlinear continuous optimization problem. To tackle this issue, this paper introduces a new hybrid method comprised of a modification of the original cuckoo search method for global optimization called improved cuckoo search (ICS) along with the Luus-Jakoola heuristics for local search. This hybrid methodology is applied to three fractal examples with 3, 4, and 26 contractive functions. Our experimental results show that the method performs very well and provides visually satisfactory solutions for the instances in our benchmark. The numerical values of the similarity index used in this work also show that the results are not optimal yet, suggesting that the method might arguably be further improved.

Keywords: Swarm intelligence · Hybrid methods
Improved cuckoo search algorithm · Luus-Jakoola heuristics
Iterated function systems · Self-similar fractal images

1 Introduction

Fractals are one of the most challenging and intriguing mathematical shapes ever defined. Basically, they are geometric figures created by repeating a simple process over and over so that it yields a self-similar pattern across different scales. Interestingly, in the case of fractals, the scale factor of this replicating pattern is not an integer number, but a real one. Such a number is called the *fractal*

L. Iliadis et al. (Eds.): AIAI 2018, IFIP AICT 519, pp. 495–506, 2018.
https://doi.org/10.1007/978-3-319-92007-8_42

dimension and it is usually larger than the topological dimension of the fractal [3,9]. Fractals have become ubiquitous objects in popular culture, particularly since the 80s of last century, owing to the technological advances in hardware and software and the widespread availability of personal computers. They are also very popular in science due to their ability to describe many growing patterns and natural structures commonly found in real-life objects: branches of trees, river networks, coastlines, mountain ranges, and so on. Furthermore, fractals have found remarkable applications in computer graphics, scientific visualization, image processing, dynamical systems, telecommunications, medicine, biology, arts, and many other fields [1,3,9,11,12].

There are several methods described in the literature to obtain fractal images. They include the Brownian motion, escape-time fractals, finite subdivision rules, L-systems, strange attractors of dynamical systems, and many others [1,3]. One of the most popular methods is the *Iterated Function Systems* (IFS), originally conceived by Hutchinson [13] and popularized by Barnsley in [1]. Roughly, an IFS consists of a finite system of contractive maps on a complete metric space. It can be proved that the Hutchinson operator over the set of all compact subsets of this space has a unique non-empty compact fixed set for the induced Hausdorff metric, called the *attractor of the IFS*. The graphical representation of this attractor is (at least approximately) a self-similar fractal image. Conversely, each self-similar fractal image can be represented by an IFS. Obtaining the parameters of such IFS is called the *IFS inverse problem*. Basically, it consists of solving an image reconstruction problem: given a fractal image, compute the IFS whose attractor approximates the input image accurately.

This IFS inverse problem has shown to be extremely difficult. In fact, the general case is still unsolved and only partial solutions have been reached so far. In this paper we propose a new approach to address this problem for the case of binary fractal images. Our methodology consists of the hybridization of a bio-inspired metaheuristics based on the cuckoo search algorithm for global optimization and a local search procedure. In particular, we consider a modification of the original cuckoo search method called the *improved cuckoo search* (ICS), which is based on the idea of allowing the method parameters to change over the generations [16]. This method is hybridized with the *Luus-Jakoola* (LJ) heuristics, a search method aimed at improving the local search step to refine the quality of the solution.

The structure of this paper is as follows: Sect. 2 introduces the basic concepts and definitions about the iterated function systems and the IFS inverse problem. Then, Sect. 3 describes the original and the improved cuckoo search algorithms. Our proposed method is described in detail in Sect. 4, while the experimental results are briefly discussed in Sect. 5. The paper closes with the main conclusions and some ideas about future work in the field.

2 Basic Concepts and Definitions

2.1 Iterated Function Systems

An *Iterated Function System* (IFS) is a finite set $\{\phi_i\}_{i=1,\ldots,\eta}$ of contractive maps $\phi_i : \mathbf{\Omega} \longrightarrow \mathbf{\Omega}$ defined on a complete metric space $\mathcal{M} = (\mathbf{\Omega}, \mathbf{\Psi})$, where $\mathbf{\Omega} \subset \mathbb{R}^n$ and $\mathbf{\Psi}$ is a distance on $\mathbf{\Omega}$. We refer to the IFS as $\mathcal{W} = \{\mathbf{\Omega}; \phi_i, \ldots, \phi_\eta\}$. For visualization purposes, in this paper we consider that the metric space $(\mathbf{\Omega}, \mathbf{\Psi})$ is $\mathbb{R}^2$ along with the Euclidean distance d_2, which is a complete metric space. Note, however, that our method can be applied to any other complete metric space of any dimension without further modifications. In this two-dimensional case, the affine transformations ϕ_κ are of the form:

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

or equivalently: $\mathbf{\Phi}_\kappa(\mathbf{\Xi}) = \mathbf{\Theta}_\kappa.\mathbf{\Xi} + \mathbf{\Sigma}_\kappa$ where $\mathbf{\Sigma}_\kappa$ is a translation vector and $\mathbf{\Theta}_\kappa$ is a 2×2 matrix with eigenvalues $\lambda_1^\kappa, \lambda_2^\kappa$ such that $|\lambda_j^\kappa| < 1$. In fact, $\mu_\kappa = |det(\mathbf{\Theta}_\kappa)| < 1$ meaning that ϕ_κ shrinks distances between points. Let us now define a transformation called the *Hutchinson operator*, Υ, on the compact subsets of $\mathbf{\Omega}$, $\mathcal{H}(\mathbf{\Omega})$, by:

$$\Upsilon(\mathcal{B}) = \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{B}) \tag{2}$$

with $\mathcal{B} \in \mathcal{H}(\mathbf{\Omega})$. If all the ϕ_κ are contractions, Υ is also a contraction in $\mathcal{H}(\mathbf{\Omega})$ with the induced Hausdorff metric [1,13]. Then, according to the fixed point theorem, Υ has a unique fixed point, $\Upsilon(\mathcal{A}) = \mathcal{A}$, called the *attractor of the IFS*.

Consider a set of probabilities $\mathcal{P} = \{\omega_1, \ldots, \omega_\eta\}$, with $\sum_{\kappa=1}^{\eta} \omega_\kappa = 1$. There exists an efficient method, known as *probabilistic algorithm*, for the generation of the attractor of an IFS. It follows from the result $\overline{\{\mathbf{\Xi_j}\}}_j = \mathcal{A}$ provided that $\mathbf{\Xi}_0 \in \mathbf{\Omega}$, where: $\mathbf{\Xi}_j = \phi_\kappa(\mathbf{\Xi}_{j-1})$ with probability $\omega_\kappa > 0$, see [2]. Picking an initial point $\mathbf{\Xi}_0$, one of the mappings in the set $\{\phi_i, \ldots, \phi_\eta\}$ is chosen at random using the weights $\{\omega_1, \ldots, \omega_\eta\}$. The selected map is then applied to generate a new point, and the same process is repeated again with the new point and so on. As a result of this stochastic iterative process, we obtain a sequence of points that converges to the fractal as the number of points increases [1,10].

2.2 The Collage Theorem

The *Collage Theorem* basically says that any digital image $\mathcal{I}$ can be approximated through an IFS [1,10]. In particular, it states that given a non-empty compact subset $\mathcal{I} \in \mathcal{H}(\mathbf{\Omega})$, the Hausdorff metric $H(.,.)$, a non-negative real threshold value $\epsilon \geq 0$, and an IFS $\mathcal{W} = \{\mathbf{\Omega}; \phi_i, \ldots, \phi_\eta\}$ on $\mathbf{\Omega}$ with contractivity factor $0 < s < 1$ (the maximum of the contractivity factors s_κ of maps ϕ_κ), if $H(\mathcal{I}, \Upsilon(\mathcal{I})) = H\left(\mathcal{I}, \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{I})\right) \leq \epsilon$ then $H(\mathcal{I}, \mathcal{A}) \leq \dfrac{\epsilon}{1-s}$, where $\mathcal{A}$ is the attractor of the IFS. That is: $H(\mathcal{I}, \mathcal{A}) \leq \dfrac{1}{1-s} H\left(\mathcal{I}, \bigcup_{\kappa=1}^{\eta} \phi_\kappa(\mathcal{I})\right)$.

2.3 The IFS Inverse Problem

Suppose that we are given an initial fractal image $\mathcal{F}^{\square}$. The *Collage Theorem* says that it is possible to obtain an IFS $\mathcal{W}$ whose attractor has a graphical representation $\mathcal{F}^{\blacksquare}$ that approximates $\mathcal{F}^{\square}$ accurately according to a similarity function $\mathcal{S}$, which measures the graphical distance between $\mathcal{F}^{\square}$ and $\mathcal{F}^{\blacksquare}$ [1]. Note that once $\mathcal{W}$ is computed, $\mathcal{F}^{\blacksquare} = \Upsilon(I^{\square})$ for any (not necessarily fractal) initial image $I^{\square}$. Mathematically, this means that we have to solve the optimization problem:

$$\underset{\{\boldsymbol{\Theta}_\kappa, \boldsymbol{\Sigma}_\kappa, \omega_\kappa\}_{\kappa=1,\ldots,\eta}}{minimize} \quad \mathcal{S}\left(\mathcal{F}^{\square}, \Upsilon(I^{\square})\right) \tag{3}$$

The problem (3) is a continuous constrained optimization problem, because all free variables in $\{\boldsymbol{\Theta}_\kappa, \boldsymbol{\Sigma}_\kappa, \omega_\kappa\}_i$ are real-valued and must satisfy the condition that the corresponding functions ϕ_κ have to be contractive. It is also a multimodal problem, since there can be several global or local minima of the similarity function. The problem is so difficult that only partial solutions have been reported so far, but the general problem still remains unsolved to a large extent. In this paper we address this problem by using a hybrid approach based on the cuckoo search algorithm described in next paragraphs.

3 The Cuckoo Search Algorithms

3.1 Original Cuckoo Search (CS)

The *Cuckoo search* (CS) is a powerful metaheuristic algorithm originally proposed by Yang and Deb in 2009 [18]. Since then, it has been successfully applied to difficult optimization problems [5,14,17,19]. The algorithm is inspired by the obligate interspecific brood-parasitism of some cuckoo species that lay their eggs in the nests of host birds of other species to escape from the parental investment in raising their offspring and minimize the risk of egg loss to other species, as the cuckoos can distributed their eggs amongst a number of different nests.

This interesting and surprising breeding behavioral pattern is the metaphor of the cuckoo search metaheuristic approach for solving optimization problems. In the cuckoo search algorithm, the eggs in the nest are interpreted as a pool of candidate solutions of an optimization problem while the cuckoo egg represents a new coming solution. The ultimate goal of the method is to use these new (and potentially better) solutions associated with the parasitic cuckoo eggs to replace the current solution associated with the eggs in the nest. This replacement, carried out iteratively, will eventually lead to a very good solution of the problem.

In addition to this representation scheme, the CS algorithm is also based on three idealized rules [18,19]:

1. Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
2. The best nests with high quality of eggs (solutions) will be carried over to the next generations;

Table 1. Cuckoo search algorithm via Lévy flights as originally proposed in [18,19].

Algorithm. Cuckoo Search via Lévy Flights

begin
Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_D)^T$
Generate initial population of n host nests $\mathbf{x}_i$ $(i = 1, 2, \ldots, n)$
while $(t < MaxGeneration)$ or (stop criterion)
Get a cuckoo (say, i) randomly by Lévy flights
Evaluate its fitness F_i
Choose a nest among n (say, j) randomly
if $(F_i > F_j)$
Replace j by the new solution
end
A fraction (p_a) of worse nests are abandoned and new ones are built via Lévy flights
Keep the best solutions (or nests with quality solutions)
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

3. The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location. For simplicity, this assumption can be approximated by a fraction p_a of the n nests being replaced by new nests (with new random solutions at new locations).

The basic steps of the CS algorithm are summarized in the pseudocode shown in Table 1. Basically, the CS algorithm starts with an initial population of n host nests and it is performed iteratively. The initial values of the jth component of the ith nest are determined by the expression $x_i^j(0) = rand.(up_i^j - low_i^j) + low_i^j$, where up_i^j and low_i^j represent the upper and lower bounds of that jth component, respectively, and $rand$ represents a standard uniform random number on the interval $(0, 1)$. With this choice, the initial values are within the search space domain. These boundary conditions are also controlled in each iteration step.

For each iteration t, a cuckoo egg i is selected randomly and new solutions $\mathbf{x}_i^{t+1}$ are generated by using the Lévy flight. According to the original creators of the method, the strategy of using Lévy flights is preferred over other simple random walks because it leads to better overall performance of the CS. The general equation for the Lévy flight is given by:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \oplus levy(\lambda) \tag{4}$$

where t indicates the number of the current generation, and $\alpha > 0$ indicates the step size, which should be related to the scale of the particular problem under study. The symbol $\oplus$ is used in Eq. (4) to indicate the entry-wise multiplication. Note that Eq. (4) is essentially a Markov chain, since next location at generation $t + 1$ only depends on the current location at generation t and a transition probability, given by the first and second terms of Eq. (4), respectively. This transition probability is modulated by the Lévy distribution as:

$$levy(\lambda) \sim t^{-\lambda}, \qquad (1 < \lambda \leq 3) \tag{5}$$

which has an infinite variance with an infinite mean. From the computational standpoint, the generation of random numbers with Lévy flights is comprised of two steps: firstly, a random direction according to a uniform distribution is chosen; then, the generation of steps following the chosen Lévy distribution is carried out. The authors suggested to use the Mantegna's algorithm for symmetric distributions (see [19] for details), which computes the factor:

$$\hat{\phi} = \left(\frac{\Gamma(1+\hat{\beta}).sin\left(\frac{\pi.\hat{\beta}}{2}\right)}{\Gamma\left(\left(\frac{1+\hat{\beta}}{2}\right).\hat{\beta}.2^{\frac{\hat{\beta}-1}{2}}\right)} \right)^{\frac{1}{\hat{\beta}}} \tag{6}$$

where Γ denotes the Gamma function and $\hat{\beta} = \dfrac{3}{2}$ in the original implementation by Yang and Deb [19]. This factor is used in Mantegna's algorithm to compute the step length ς as: $\varsigma = \frac{u}{|v|^{\frac{1}{\hat{\beta}}}}$, where u and v follow the normal distribution of zero mean and deviation σ_u^2 and σ_v^2, respectively, where σ_u obeys the Lévy distribution given by Eq. (6) and $\sigma_v = 1$. Then, the stepsize ζ is computed as $\zeta = 0.01\,\varsigma\,(\mathbf{x} - \mathbf{x}_{best})$. Finally, $\mathbf{x}$ is modified as: $\mathbf{x} \leftarrow \mathbf{x} + \zeta.\boldsymbol{\Delta}$ where $\boldsymbol{\Delta}$ is a random vector of the dimension of the solution $\mathbf{x}$ and that follows the normal distribution $N(0, 1)$. The CS method then evaluates the fitness of the new solution and compares it with the current one. In case the new solution brings better fitness, it replaces the current one. On the other hand, a fraction of the worse nests (according to the fitness) are abandoned and replaced by new solutions so as to increase the exploration of the search space looking for more promising solutions. The rate of replacement is given by the probability p_a, a parameter of the model that has to be tuned for better performance. Moreover, for each iteration step, all current solutions are ranked according to their fitness and the best solution reached so far is stored as the vector $\mathbf{x}_{best}$.

3.2 Improved Cuckoo Search (ICS)

The *improved cuckoo search* (ICS) method was proposed in [16] to enhance the performance of the original CS. It is based on the idea of allowing its parameters p_a and α to change over the generations, as opposed to their fixed values in the original CS. In ICS the parameter p_a is modified as:

$$p_a^t = p_{a_M} - \frac{p_{a_M} - p_{a_m}}{\Lambda}\, t \tag{7}$$

where the subscripts M and m are used to indicate the maximum and minimum values of the parameter respectively, and Λ indicates the total number of iterations. According to Eq. (7), the parameter p_a is now decreased linearly with the number of iterations from a maximum value p_{a_M} until a minimum one, p_{a_m}. At early iterations, its value is high to enforce the diversity of solutions in the algorithm. This diversity is decreasing over the time so as to intensify the search using the best candidates of the population in final iterations for a better fine-tuning of the solutions. The parameter α, also assumed constant in the CS, is primarily used to promote exploration of the search space. Therefore, it makes sense to modify it dynamically starting from a high value, α_M, to perform a extensive exploration and gradually reducing it until a low value, α_m, to promote exploitation and eventually homing into the optimum, in a rather similar way to the inertia weight in PSO. Consequently, it is modified as:

$$\alpha^t = \alpha_M \, Exp\left(\frac{Ln\left(\frac{\alpha_m}{\alpha_M}\right)}{\Lambda} t\right) \tag{8}$$

4 Proposed Approach

4.1 Hybrid ICS with Luus-Jakoola Heuristics

To address the IFS inverse problem described in Sect. 2.2, we propose a new hybrid scheme for proper balance between exploration and exploitation. Firstly, we consider the improved cuckoo search method for global optimization described in Sect. 3.2. This modified scheme is improved by its hybridization with a local search procedure. In particular, we apply the *Luus-Jaakola* (LJ) method, a gradient-free heuristics firstly proposed in [15] to solve nonlinear programming problems. LJ starts with an initialization step, where random uniform values are chosen within the search space by computing the upper and lower bounds for each dimension. Then, a random uniform value in-between is sampled for each component. This value is added to the current position of the potential solution to generate a new candidate solution, which replaces the current one only when the fitness improves; otherwise, the sampling space is multiplicatively decreased by a factor (usually of value 95%, but other values can also be used). In practice, we found that it is better to consider a self-adaptive size for this factor, with the effect of speeding up the convergence to the steady state. This process is repeated iteratively. With each iteration, the neighborhood of the point decreases, so the procedure eventually collapses to a point.

4.2 Application to the IFS Inverse Problem

Given a 2D self-similar binary fractal image $\mathcal{I}^{\square}$ comprised of η functions ϕ_κ, we apply our hybrid method to solve the IFS inverse problem. We consider an

initial population of χ individuals $\{\mathcal{C}_i\}_{i=1,\ldots,\chi}$, where each individual $\mathcal{C}_i = \{\mathcal{C}_i^\kappa\}_\kappa$ is a collection of η real-valued vectors $\mathcal{C}_\kappa^i$ of the free variables of Eq. (1), as:

$$\mathcal{C}_i^\kappa = (\theta_{1,1}^{\kappa,i}, \theta_{1,2}^{\kappa,i}, \theta_{2,1}^{\kappa,i}, \theta_{2,2}^{\kappa,i} | \sigma_1^{\kappa,i}, \sigma_2^{\kappa,i} | \omega_\kappa^i) \tag{9}$$

These individuals are initialized with uniform random values in $[-1, 1]$ for the variables in $\mathbf{\Theta}_\kappa$ and $\mathbf{\Sigma}_\kappa$, and in $[0, 1]$ for the ω_κ^i, such that $\sum_{\kappa=1}^{\eta} \omega_\kappa^i = 1$. After this initialization step, we compute the contractive factors μ_κ and reinitialize all functions ϕ_κ with $\mu_\kappa \geq 1$ to ensure that only contractive functions are included in the initial population. Before applying our method, we also need to define a suitable fitness function. In this paper we use the Hamming distance: the fractal images are stored as bitmap images of 0s and 1s for a given resolution defined by a mesh size parameter, m_s. Then, we divide the number of different values between the original and the reconstructed matrices by the total number of boxes in the image. This yields the *normalized similarity error rate* index between both images, denoted by $|\mathcal{S}|_{\square}^{\blacksquare}$, the fitness function used in this work.

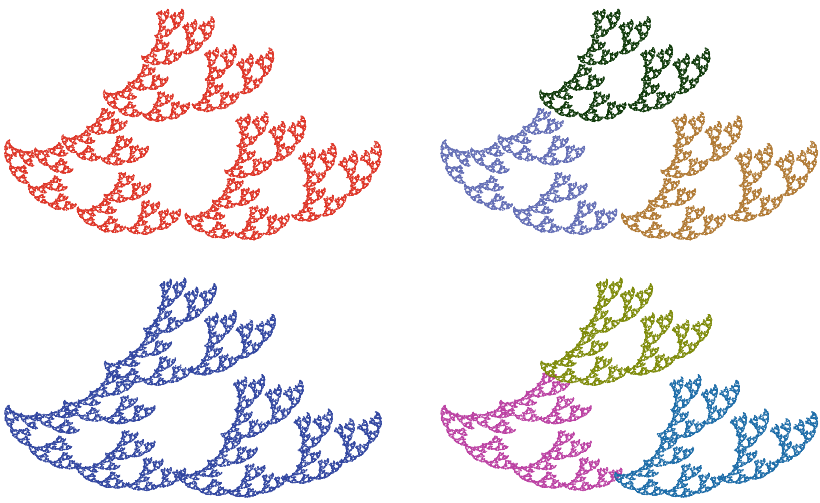

Fig. 1. Example of the *rotated triangle* fractal: (left) original (top) and reconstructed image (bottom); (right) their different contractive functions with different colors. (Color figure online)

4.3 Parameter Tuning

It is well-known that the parameter tuning of metaheuristic methods is troublesome and problem-dependent. The cuckoo search is specially advantageous in this regard, as it depends on only two parameters: the population size, χ, set to $\chi = 100$, and the probability p_a, calculated taking: $p_{a_M} = 0.5$ and $p_{a_m} = 0.005$ in Eq. (7). Moreover, the method is executed for Λ iterations. In our simulations, we

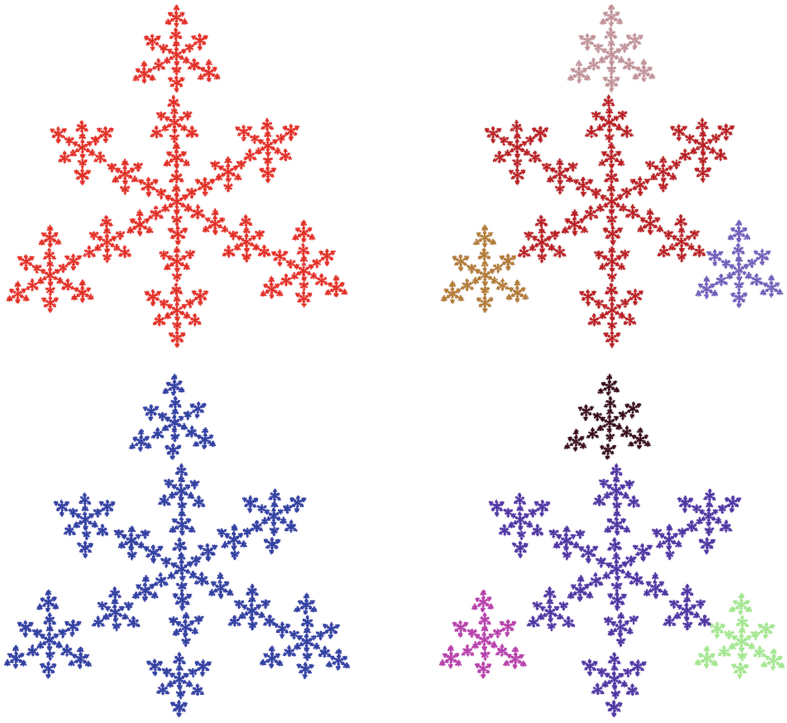

Fig. 2. Example of the *Crystal* fractal: (left) original (top) and reconstructed image (bottom); (right) their different contractive functions with different colors. (Color figure online)

found that $\Lambda = 1500$ is enough to reach convergence in mp all cases. Our hybrid method also requires to define suitable values for α_M and α_m in Eq. (8). Similar to [16], in this work they are set to 0.5 and 0.01, respectively. Finally, our method requires to define the mesh size, ν, set to $\nu = 80$ in this work.

5 Experimental Results

Our method has been applied to several examples of fractals. Only three of them are included here because of limitations of space: rotated triangle, Crystal, and the *AIAI-2018* fractal (especially created for this paper). They are shown in Figs. 1, 2 and 3, respectively. The input fractal images are displayed twice in each figure, one in red and another with one color for each individual contractive function. As the reader can see, the three fractals are comprised of 3, 4, and 26 functions, respectively. The application of our method to these input images returns the IFS minimizing the $|\mathcal{S}|^{\blacksquare}_{\square}$ index, which are then used to render the reconstructed fractal images, shown in blue and with different colors as above.

As the reader can see, although the matching is not optimal, our method captures the underlying structure of the fractal images with good visual quality.

Fig. 3. Example of the *AIAI-2018* fractal (from top to bottom): original image in red, original image with different colors for each contractive function, reconstructed image in blue, and reconstructed image with different colors for each contractive function. (Color figure online)

This is a remarkable result because our initial population is totally random, meaning that their corresponding images at early generations are all totally different from the given image. Our method is able to obtain a final image that replicates well the input image and is also a self-similar fractal for all instances in our benchmark. Our graphical results indicate that our method performs very well, as it provides visually satisfactory solutions for the IFS inverse problem.

This good visual appearance is confirmed by our numerical results, reported in Table 2. For each fractal example (in rows), the table shows (in columns): number of contractive functions η, number of free variables of the optimization problem, and best and mean value of the $|\mathcal{S}|^{\blacksquare}_{\square}$ index for 20 independent executions of the method. As the reader can see, the similarity index for the best execution yields errors ranging from 20% to 30%, depending on the example. Although the reconstructed images are perfectly recognizable in all cases, this similarity error increases with the complexity of the model, with the third example exhibiting the larger value. This result is very reasonable because this example requires to solve an optimization problem with many more variables than the other two examples. In addition, the numerical values of the similarity index used in this work indicate that the results are not optimal yet, suggesting that the method might arguably be further improved.

Table 2. Numerical results of the best and mean values of the $|\mathcal{S}|^{\blacksquare}_{\square}$ index for the examples in Figs. 1, 2 and 3 with the proposed method.

Example	η	#free variables	$\|\mathcal{S}\|^{\blacksquare}_{\square}$ (best)	$\|\mathcal{S}\|^{\blacksquare}_{\square}$ (mean)
Rotated triangle fractal	3	21	0.1952	0.2113
Crystal fractal	4	28	0.2038	0.2194
AIAI-2018 fractal	26	182	0.2974	0.3125

All computations in this paper have been performed on a 2.6 GHz. Intel Core i7 processor with 16 GB of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab version 2015a* and using the numerical libraries for fractals in [4,6–8]. Regarding the CPU times, they depend on the complexity of the model and its number of contractive functions. In general, we noticed that the method is slow and time-consuming. For illustration, each single execution takes about 25–45 min for the first two examples, and more than 1 h for the third one.

6 Conclusions and Future Work

In this paper we introduced a new hybrid method to solve the following optimization problem: given any binary self-similar fractal image, the goal is to determine an IFS whose attractor is a good approximation of this input image. This problem, called the IFS inverse problem, is known to be a very difficult multivariate nonlinear continuous optimization problem. In this new hybrid method, a modification of the original cuckoo search method for global optimization called improved cuckoo search (ICS) is coupled with the Luus-Jakoola heuristics for local search. This hybrid methodology is applied to three fractal examples: *rotated triangle*, *crystal*, and the *AIAI-2018* fractal. The method replicates the input images very well, yielding visually satisfactory results in all cases.

The numerical results show however that our final solutions are not optimal yet, so the method might be improved in several ways. On one hand, we want to modify our fitness function to obtain a better measure of the quality of the reconstructed fractal. On the other hand, we would like to obtain automatically the optimal value of the number of contractive functions for the IFS. We also wish to extend our results to the cases of images that are neither binary nor self-similar. Reducing our CPU times is also one of our future goals in the field.

Acknowledgements. This research work has received funding from the project PDE-GIR of the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 778035, the Spanish Ministry of Economy and Competitiveness (Computer Science National Program) under grant #TIN2017-89275-R of the Agencia Estatal de Investigación and European Funds FEDER (AEI/FEDER, UE), and the project #JU12, jointly supported by public body SODERCAN and European Funds FEDER (SODERCAN/FEDER UE). We also thank Toho University and the University of Cantabria for their support for this research work.

References

1. Barnsley, M.F.: Fractals Everywhere, 2nd edn. Academic Press, San Diego (1993)
2. Elton, J.H.: An ergodic theorem for iterated maps. Ergodic Theory Dynam. Syst. **7**, 481–488 (1987)
3. Falconer, K.: Fractal Geometry: Mathematical Foundations and Applications, 2nd edn. Wiley, Chichester (2003)
4. Gálvez, A.: IFS Matlab generator: a computer tool for displaying IFS fractals. In: Proceedings of ICCSA 2009, pp. 132–142. IEEE CS Press, Los Alamitos (2009)
5. Gálvez, A., Iglesias, A.: Cuckoo search with Lévy flights for weighted Bayesian energy functional optimization in global-support curve data fitting. Sci. W. J. **2014**, 11 (2014). Article ID 138760
6. Gálvez, A., Iglesias, A., Takato, S.: Matlab-based KETpic add-on for generating and rendering IFS fractals. In: Ślęzak, D., Kim, T., Chang, A.C.-C., Vasilakos, T., Li, M.C., Sakurai, K. (eds.) FGCN 2009. CCIS, vol. 56, pp. 334–341. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10844-0_40
7. Gálvez, A., Iglesias, A., Takato, S.: KETpic Matlab binding for efficient handling of fractal images. Int. J. Future Gener. Commun. Netw. **3**(2), 1–14 (2010)
8. Gálvez, A., Kitahara, K., Kaneko, M.: *IFSGen4*LaTeX: interactive graphical user interface for generation and visualization of iterated function systems inLaTeX. In: Hong, H., Yap, Chee (eds.) ICMS 2014. LNCS, vol. 8592, pp. 554–561. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_84
9. Gutiérrez, J.M., Iglesias, A.: A mathematica package for the analysis and control of chaos in nonlinear systems. Comput. Phys. **12**(6), 608–619 (1998)
10. 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)
11. Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A., Burgos, J.D., Moreno, P.A.: Analyzing the multifractal structure of DNA nucleotide sequences. In: Chaos and Noise in Biology and Medicine, vol. 7, pp. 315–319. World Scientific, Singapore (1998)
12. Gutiérrez, J.M., Iglesias, A., Rodríguez, M.A., Rodríguez, V.J.: Generating and rendering fractal images. Mathematica J. **7**(1), 6–13 (1997)
13. Hutchinson, J.E.: Fractals and self similarity. Indiana Univ. Math. J. **30**(5), 713–747 (1981)
14. Iglesias, A., Gálvez, A.: Cuckoo search with Lévy flights for reconstruction of outline curves of computer fonts with rational Bézier curves. In: Proceedings of Congress on Evolutionary Computation-CEC 2016. IEEE CS Press, Los Alamitos (2016)
15. Luus, R., Jaakola, T.H.I.: Optimization by direct search and systematic reduction of the size of search region. Am. Inst. Chem. Eng. J. (AIChE) **19**(4), 760–766 (1973)
16. Valian, E., Tavakoli, S., Mohanna, S., Hahgi, A.: Improved cuckoo search for reliability optimization problems. Comput. Industr. Eng. **64**, 459–468 (2013)
17. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, Frome (2010)
18. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 210–214. IEEE Press, New York (2009)
19. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Modelling Numer. Optim. **1**(4), 330–343 (2010)