Numerical Algorithms (2020) 83:1399–1413
https://doi.org/10.1007/s11075-019-00730-w

ORIGINAL PAPER

A new algorithm that generates the image of the attractor of a generalized iterated function system

Radu Miculescu[1] · Alexandru Mihail[2] · Silviu-Aurelian Urziceanu[3]

Received: 27 July 2018 / Accepted: 15 May 2019 / Published online: 28 May 2019

Abstract

We provide a new algorithm (called the grid algorithm) designed to generate the image of the attractor of a generalized iterated function system on a finite dimensional space and we compare it with the deterministic algorithm regarding generalized iterated function systems presented by Jaros et al. (Numer. Algorithms **73**, 477–499, 2016).

Keywords Generalized infinite iterated function system (GIFS) · Attractor · Deterministic algorithm · Grid algorithm

Mathematics Subject Classification (2010) Primary 28A80; Secondary 37C70 · 41A65 · 65S05 · 65P99

✉ Radu Miculescu
radu.miculescu@unitbv.ro

Alexandru Mihail
mihail_alex@yahoo.com

Silviu-Aurelian Urziceanu
fmi_silviu@yahoo.com

[1] Faculty of Mathematics and Computer Science, Transilvania University of Braşov, Iuliu Maniu Street, nr. 50, 500091, Braşov, Romania

[2] Faculty of Mathematics and Computer Science, University of Bucharest, Romania, Academiei Street 14, 010014, Bucharest, Romania

[3] Faculty of Mathematics and Computer Science, University of Piteşti, Romania, Târgul din Vale 1, 110040, Piteşti, Argeş, Romania

1 Introduction

As part of the effort to extend the classical theory of iterated function systems due to J. Hutchinson (see [2]), R. Miculescu and A. Mihail introduced the concept of generalized iterated function system (see [7] and [9]) which was obtained by considering contractions from X^p into X rather than contractions from X into itself (here (X, d) is a metric space and p is a natural number). Sufficient conditions for the existence and uniqueness of the attractor of a generalized iterated function system (for short GIFS) $\mathcal{F} = ((X, d), (f_i)_{i\in\{1,2,...,L\}})$, an upper bound for the Hausdorff–Pompeiu distance between the attractors of two such GIFSs, an upper bound for the Hausdorff–Pompeiu distance between the attractor of such a GIFS and an arbitrary compact set of X have been provided and the continuous dependence of the attractor on the functions f_i was proved. In the last years, this concept has been intensively studied. Let us mention some lines of research regarding this subject:

In [15], F. Strobin and J. Swaczyna extended the concept of GIFS by using weaker types of generalized contractions which are similar to those introduced by F. Browder (see [1]) or J. Matkowski (see [5]). In [14], Strobin emphasized the fact that the set of the attractors generated by GIFSs is larger than the one generated by iterated function systems. Other related topics can be found in [4, 6, 8, 10–13] and [16].

Moreover, in [3], Strobin and his collaborators provided four algorithms which generate images of attractors of GIFSs, one of them being the deterministic algorithm for GIFSs (a counterpart of the classical deterministic algorithm for iterated function systems). Note that in [3] one can also find a list of papers dealing with algorithms generating images of the attractors of iterated function systems.

In this paper, we present another algorithm (called the grid algorithm) allowing the generation of images of the attractors of GIFSs on finite dimensional spaces and we compare it with the deterministic algorithm for GIFSs. The deterministic algorithm for GIFSs consists in choosing a finite set of points and applying to this set each of the constitutive functions of the system obtaining in this way a new finite set of points. To each of these new points, we apply again each of the constitutive functions of the system. Continuing the procedure described above, we approach the attractor. The main idea of the grid algorithm is to simplify the deterministic algorithm by dividing, at each step, the space that we are working with into small pieces and to choose for each such piece just one point.

2 Preliminaries

Given a metric space (X, d), we adopt the following notation:

$$P_{cp}(X) \overset{not}{=} \{K \subseteq X \mid K \text{ is non-empty and compact}\}.$$

For $K_1, K_2 \in P_{cp}(X)$, we consider:

$$d(K_1, K_2) \overset{def}{=} \sup_{x\in K_1} d(x, K_2),$$

where $d(x, K_2) \overset{def}{=} \inf\limits_{y \in K_2} d(x, y)$.

The function $h : P_{cp}(X) \times P_{cp}(X) \to [0, \infty)$ given by:

$$h(K_1, K_2) = \max\{d(K_1, K_2), d(K_2, K_1)\},$$

for every $K_1, K_2 \in P_{cp}(X)$, turns out to be a metric which is called the Hausdorff-Pompeiu metric.

If (X, d) is complete, then $(P_{cp}(X), h)$ is complete.

Given a metric space (X, d) and $p \in \mathbb{N}^*$, by X^p we denote the Cartesian product of p copies of X. We endow X^p with the maximum metric $d_{\max}$ defined by:

$$d_{\max}((x_1, ..., x_p), (y_1, ..., y_p)) = \max\{d(x_1, y_1), ..., d(x_p, y_p)\},$$

for all $(x_1, ..., x_p), (y_1, ..., y_p) \in X^p$.

Definition 2.1 A generalized iterated function system (of order p) is a pair $\mathcal{F} = ((X, d), (f_i)_{i \in \{1,2,...,L\}})$, where (X, d) is a metric space, $p, L \in \mathbb{N}^*$ and $f_i : X^p \to X$ is contraction for each $i \in \{1, ..., L\}$. The function $F_{\mathcal{F}} : (P_{cp}(X))^p \to P_{cp}(X)$, described by:

$$F_{\mathcal{F}}(K_1, ..., K_p) = \bigcup_{i \in \{1,...,L\}} f_i(K_1 \times ... \times K_p),$$

for all $K_1, ..., K_p \in P_{cp}(X)$, is called the fractal operator associated to $\mathcal{F}$.

We shall use the abbreviation GIFS for a generalized iterated function system.

Theorem 2.2 *(see Theorem 3.9 from* [9]*). Given a complete metric space (X, d) and a GIFS $\mathcal{F} = ((X, d), (f_i)_{i \in \{1,...,L\}})$ of order p, there exists a unique $A_{\mathcal{F}} \in P_{cp}(X)$ such that:*

$$F_{\mathcal{F}}(A_{\mathcal{F}}, ..., A_{\mathcal{F}}) = A_{\mathcal{F}}.$$

In addition, for every $K_1, ..., K_p \in P_{cp}(X)$, the sequence $(K_n)_n$ defined by

$$K_{n+p} = F_{\mathcal{F}}(K_n, ..., K_{n+p-1}),$$

for every $n \in \mathbb{N}^*$, converges, with respect to the Hausdorff-Pompeiu metric, to $A_{\mathcal{F}}$.

Definition 2.3 In the framework of the above theorem, the set $A_{\mathcal{F}}$ is called the fractal generated by $\mathcal{F}$.

Remark 2.4 (see Remark 12 from [3]). In the framework of the above definition, the function $\mathcal{G}_{\mathcal{F}} : P_{cp}(X) \to P_{cp}(X)$, described by:

$$\mathcal{G}_{\mathcal{F}}(K) = F_{\mathcal{F}}(K, ..., K) = \bigcup_{i \in \{1,...,L\}} f_i(K \times ... \times K),$$

for all $K \in P_{cp}(X)$, is a contraction on the complete metric space $(P_{cp}(X), h)$ since it has the Lipschitz constant less than or equal to $\max\{lip(f_1), ..., lip(f_L)\} < 1$.

3 The presentation of the algorithms

For $(x_1, ..., x_M) \in \mathbb{R}^M$, we shall use the following notation:

$$[(x_1, ..., x_M)] = ([x_1], ..., [x_M]),$$

where $[x]$ designates the greatest integer less than or equal to the real number x.

In the sequel, without loss of generality,

$$\mathcal{F} = (([0, D]^M, d), \{f_1, ..., f_L\}),$$

where $L, M \in \mathbb{N}$ and d is the Euclidean distance in $\mathbb{R}^M$, will be a generalized iterated function system of order $p \geq 2$ (so $f_i : ([0, D]^M)^p \to [0, D]^M$ for every $i \in \{1, ..., L\}$).

We shall use the following notation:

- $\max\{lip(f_1), ..., lip(f_L)\} \overset{not}{=} C < 1$
- $\beta = pM$.

We also consider the following functions:

- $F_{\mathcal{F}} : (P_{cp}([0, D]^M))^p \to P_{cp}([0, D]^M)$ described by

$$F_{\mathcal{F}}(K_1, ..., K_p) = f_1(K_1 \times ... \times K_p) \cup ... \cup f_L(K_1 \times ... \times K_p),$$

for all $K_1, K_p \in P_{cp}([0, D]^M)$

- $\mathcal{G}_{\mathcal{F}} : P_{cp}([0, D]^M) \to P_{cp}([0, D]^M)$ described by

$$\mathcal{G}_{\mathcal{F}}(K) = F_{\mathcal{F}}(K, K),$$

for every $K \in P_{cp}([0, D]^M)$

- $(n_k)_{k \in \mathbb{N}^*}$ a sequence of natural numbers.

For a finite set $K_0 \in P_{cp}([0, D]^M)$, we shall use the following notations:

- $$A_k \overset{not}{=} \mathcal{G}_{\mathcal{F}}^{[k]}(K_0),$$

where $k \in \mathbb{N}$

- $$\tilde{A}_k \overset{not}{=} \{\frac{D}{n_k}[\frac{n_k}{D} f_l(u_1, ..., u_p)] \mid u_1, ..., u_p \in \tilde{A}_{k-1}, l \in \{1, ..., L\}\},$$

where $k \in \mathbb{N}^*$ and $\tilde{A}_0 = K_0$

- $$\frac{D\sqrt{M}}{n_k} \overset{not}{=} \varepsilon_k,$$

where $k \in \mathbb{N}$.

Let us recall the pseudocode for the deterministic algorithm for a GIFS (see [3]):

```
Pseudocode for the deterministic algorithm for a GIFS
Read initially defined objects: constants: $L, M$, finite set, $m$ natural number: $K_0 \in P_{cp}([0, D]^M)$, mappings: $f_1, ..., f_L$, variables: $k, D_0$.
Initial values: $D_0 := K_0$.
        For $k$ from 1 to $m - 1$
            $D_1 := \mathcal{G}_{\mathcal{F}}(D_0)$
            $D_0 := D_1$.
Print $D_m$.
```

Now, let us present the pseudocode for our new algorithm.

```
Pseudocode for the grid algorithm for a GIFS
Read initially defined objects: constant: $L$, $M$, finite set, $m$ natural number: $K_0 \in P_{cp}([0, D]^M)$, mappings: $f_1, ..., f_L$, sequence: $(n_k)_k$, variables: $k, D_0$.
Initial values: $D_0 := K_0$.
    For $k$ from 1 to $m - 1$
        $D_1 := \{\frac{D}{n_k}[\frac{n_k}{D} f_l(u_1, ..., u_p)] \mid u_1, ..., u_p \in D_0, l \in \{1, ..., L\}\}$
        $D_0 := D_1$.
Print $D_m$.
```

4 The complexity of the algorithms

By x_k, we denote the number of points computed at the step k of the deterministic algorithm and by y_k the number of points computed up to the step k of the grid algorithm.

A. The case of the deterministic algorithm

We have $x_{k+1} \leq L(x_k)^p$, so, with the notation $z_k \overset{not}{=} \ln x_k$, we obtain $z_{k+1} \leq \ln L + pz_k$ for every $k \in \mathbb{N}$. Therefore $z_k \leq \frac{p^k-1}{p-1} \ln L + p^k z_0$, i.e.:

$$x_k \leq \frac{1}{L^{\frac{1}{p-1}}}(x_0 L^{\frac{1}{p-1}})^{p^k}, \tag{1}$$

for every $k \in \mathbb{N}$.

Note that, according to Remark 2.4, we have $h(A_k, A_{\mathcal{F}}) \leq \frac{h(A_0,A_1)}{1-C} C^k$ for every $k \in \mathbb{N}$. Therefore, in order to be sure that A_k approximates the attractor $A_{\mathcal{F}}$ with accuracy $\varepsilon \frac{h(A_0,A_1)}{1-C}$, we need $k > \log_{C^{-1}}(\varepsilon^{-1})$. Hence, based on (1), the quantity $\frac{1}{L^{\frac{1}{p-1}}}(x_0 L^{\frac{1}{p-1}})^{p^{\log_{C^{-1}}(\varepsilon^{-1})}} = \frac{1}{L^{\frac{1}{p-1}}}(x_0 L^{\frac{1}{p-1}})^{(\frac{1}{\varepsilon})^{\log_{C^{-1}}(p)}}$ describes the number of points that we have to compute in order to be sure that A_k is an approximation of $A_{\mathcal{F}}$ with an error less than $\varepsilon \frac{h(A_0,A_1)}{1-C}$.

Conclusion The complexity of the deterministic algorithm is described by the function $\mathcal{C}_c : (0, \infty) \to \mathbb{R}$ given by:

$$\mathcal{C}_c(\varepsilon) = (x_0 L^{\frac{1}{p-1}})^{(\frac{1}{\varepsilon})^{\log_{\frac{1}{C}}(p)}},$$

for every $\varepsilon > 0$.

B. The case of the grid algorithm

Remark 4.1 Since $y_{k+1} \leq L(n_k)^{\beta}$ for every $k \in \mathbb{N}^*$, up to the step N, we have to compute $L\sum\limits_{k=1}^{N}(n_k)^{\beta} = L(D\sqrt{M})^{\beta}\sum\limits_{k=1}^{N}(\frac{1}{\varepsilon_k})^{\beta}$ points.

Remark 4.2 We have

$$h(\tilde{A}_k, \mathcal{G}_{\mathcal{F}}(\tilde{A}_{k-1})) \leq \varepsilon_k,$$

for every $k \in \mathbb{N}^*$.

Remark 4.3 We have

$$h(\tilde{A}_0, A_{\mathcal{F}}) \leq diam([0, D]^M) = D\sqrt{M}\,.$$

As the inequality:

$$h(\tilde{A}_k, A_{\mathcal{F}}) \leq h(\tilde{A}_k, \mathcal{G}_{\mathcal{F}}(\tilde{A}_{k-1})) + h(\mathcal{G}_{\mathcal{F}}(\tilde{A}_{k-1}), \mathcal{G}_{\mathcal{F}}(A_{\mathcal{F}})) \leq$$
$$\overset{\text{Remarks 2.4 and 4.2}}{\leq} \varepsilon_k + Ch(\tilde{A}_{k-1}, A_{\mathcal{F}}),$$

is valid for every $k \in \mathbb{N}^*$, we get:

$$h(\tilde{A}_k, A_{\mathcal{F}}) \leq \varepsilon_k + C\varepsilon_{k-1} + C^2\varepsilon_{k-2} + ... + C^{k-2}\varepsilon_2 + C^{k-1}\varepsilon_1 + C^k h(\tilde{A}_0, A_{\mathcal{F}}),$$

so, taking into account Remark 4.3, we obtain:

$$h(\tilde{A}_k, A_{\mathcal{F}}) \leq \varepsilon_k + C\varepsilon_{k-1} + C^2\varepsilon_{k-2} + ... + C^{k-2}\varepsilon_2 + C^{k-1}\varepsilon_1 + C^k D\sqrt{M},$$

for every $k \in \mathbb{N}^*$. Consequently, we arrive to the following problem: given a fixed natural number N and $\varepsilon > 0$ such that $\frac{\varepsilon}{C^N} - D\sqrt{M} > 0$, find the minimum of the function $f : [0, \infty)^N \to [0, \infty)$, given by:

$$f(\varepsilon_1, ..., \varepsilon_N) = \sum_{k=1}^{N}(\frac{1}{\varepsilon_k})^{\beta},$$

for every $\varepsilon_1, ..., \varepsilon_N \in [0, \infty)$, with the constraint:

$$\varepsilon_N + C\varepsilon_{N-1} + C^2\varepsilon_{N-2} + ... + C^{N-2}\varepsilon_2 + C^{N-1}\varepsilon_1 + C^N D\sqrt{M} = \varepsilon.$$

We adopt the following notations:

- $t \overset{not}{=} C^{-\frac{\beta}{\beta+1}N} - 1$

- $K_1 \overset{not}{=} \frac{C^{\frac{1}{\beta+1}}-C}{C^{\frac{1}{\beta+1}}} = 1 - C^{\frac{\beta}{\beta+1}}$
- $K_2 \overset{not}{=} K_1^{-\beta-1}$
- $K_3 \overset{not}{=} K_2\varepsilon^{-\beta}$
- $a \overset{not}{=} \frac{D\sqrt{M}}{\varepsilon}$
- $y \overset{not}{=} \frac{1}{C^N}$.

Since we are going to use the method of Lagrange multipliers, we consider the function $F = f + \lambda g$, where $\lambda \in \mathbb{R}$ and the function $g : [0, \infty)^N \to [0, \infty)$ is given by:

$$g(\varepsilon_1, ..., \varepsilon_N) = \varepsilon_N + C\varepsilon_{N-1} + ... + C^{N-2}\varepsilon_2 + C^{N-1}\varepsilon_1 + C^N D\sqrt{M} - \varepsilon,$$

for every $\varepsilon_1, ..., \varepsilon_N \in [0, \infty)$. The equation $\frac{\partial F}{\partial \varepsilon_k} = 0$, i.e. $-\beta(\varepsilon_k)^{-\beta-1} + \lambda C^{N-k} = 0$, has the solution:

$$\varepsilon_k^0 = k_N C^{\frac{k}{\beta+1}}, \tag{1}$$

for every k, where $k_N = \frac{1}{C^{\frac{N}{\beta+1}}}(\frac{\beta}{\lambda})^{\frac{1}{\beta+1}}$. As $g(\varepsilon_1^0, ..., \varepsilon_N^0) = 0$, we get:

$$k_N(C^{\frac{N}{\beta+1}} + C^{1+\frac{N-1}{\beta+1}} + C^{2+\frac{N-2}{\beta+1}} + ... + C^{N-2+\frac{2}{\beta+1}} + C^{N-1+\frac{1}{\beta+1}}) = \varepsilon - C^N D\sqrt{M},$$

i.e.:

$$k_N C^{\frac{N}{\beta+1}}(1 + C^{\frac{\beta}{\beta+1}} + C^{2\frac{\beta}{\beta+1}} + ... + C^{(N-2)\frac{\beta}{\beta+1}} + C^{(N-1)\frac{\beta}{\beta+1}}) = \varepsilon - C^N D\sqrt{M},$$

so $k_N C^{\frac{N}{\beta+1}} \frac{(C^{\frac{\beta}{\beta+1}})^N - 1}{C^{\frac{\beta}{\beta+1}} - 1} = \varepsilon - C^N D\sqrt{M}$, which implies $k_N \frac{C^N - C^{\frac{N}{\beta+1}}}{C - C^{\frac{1}{\beta+1}}} = \frac{\varepsilon - C^N D\sqrt{M}}{C^{\frac{1}{\beta+1}}}$. The last equality takes the form $k_N \frac{C^{-\frac{\beta}{\beta+1}N} - 1}{C^{\frac{1}{\beta+1}} - C} = \frac{\frac{\varepsilon}{C^N} - D\sqrt{M}}{C^{\frac{1}{\beta+1}}}$. Thus, we obtain:

$$k_N = \frac{K_1}{t}(\frac{\varepsilon}{C^N} - D\sqrt{M}). \tag{2}$$

We have:

$$f(\varepsilon_1^0, ..., \varepsilon_N^0) = \sum_{k=1}^{N}(\varepsilon_k^0)^{-\beta} \overset{(1)}{=} (k_N)^{-\beta}\sum_{k=1}^{N} C^{-\frac{\beta}{\beta+1}k} = (k_N)^{-\beta} C^{-\frac{\beta}{\beta+1}} \frac{(C^{-\frac{\beta}{\beta+1}})^N - 1}{C^{-\frac{\beta}{\beta+1}} - 1} =$$

$$\overset{(2)}{=} t^\beta K_1^{-\beta}(\frac{\varepsilon}{C^N} - D\sqrt{M})^{-\beta} \frac{t}{1 - C^{\frac{\beta}{\beta+1}}} = t^{\beta+1}(\frac{\varepsilon}{C^N} - D\sqrt{M})^{-\beta} \frac{K_1^{-\beta}}{K_1} =$$

$$= t^{\beta+1}(\frac{\varepsilon}{C^N} - D\sqrt{M})^{-\beta} K_1^{-\beta-1} = K_2(\frac{\varepsilon}{C^N} - D\sqrt{M})^{-\beta}(C^{-\frac{\beta}{\beta+1}N} - 1)^{\beta+1}.$$

Therefore, the last equality can be written as:

$$f(\varepsilon_1^0, ..., \varepsilon_N^0) = K_3(y^{\frac{\beta}{\beta+1}} - 1)^{\beta+1}(y - a)^{-\beta}. \tag{3}$$

As the right-hand side of (3) gives us the optimal number of points that we have to compute, after N steps, in order to approximate $A_{\mathcal{F}}$ by $\overset{\sim}{A_k}$ with an error not greater than ε, we need to find the minimum value of the function $h : (a, \infty) \to \mathbb{R}$ given by:

$$h(y) = K_3(y^{\frac{\beta}{\beta+1}} - 1)^{\beta+1}(y-a)^{-\beta},$$

for every $y \in (a, \infty)$. One can easily see that:

$$h^{'}(y) = K_3\beta(y^{\frac{\beta}{\beta+1}} - 1)^{\beta}(y-a)^{-\beta-1}(1 - ay^{-\frac{1}{\beta+1}}),$$

for every $y \in (a, \infty)$. As we can suppose that $a > 1$ (since we are interested in the case when ε is small), $\lim\limits_{y\to\infty} h(y) = K_3$ and $\lim\limits_{\substack{y\to a\\ y>a}} h(y) = \infty$, we conclude that h attains its minimum at $a^{\beta+1}$ and the value of the minimum is:

$$h(a^{\beta+1}) = K_3(a^{\beta} - 1)^{\beta+1}(a^{\beta+1} - a)^{-\beta} =$$

$$= K_3\frac{a^{\beta}-1}{a^{\beta}} = \varepsilon^{-\beta}(1 - C^{\frac{\beta}{\beta+1}})^{-\beta-1}\frac{(\frac{D\sqrt{M}}{\varepsilon})^{\beta} - 1}{(\frac{D\sqrt{M}}{\varepsilon})^{\beta}},$$

so

$$\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}} \frac{h(a^{\beta+1})}{(1 - C^{\frac{\beta}{\beta+1}})^{-\beta-1}(\frac{1}{\varepsilon})^{\beta}} = 1.$$

Conclusion The complexity of the grid algorithm is described by the function $\mathcal{C}_g : (0, \infty) \to \mathbb{R}$ given by:

$$\mathcal{C}_g(\varepsilon) = (1 - C^{\frac{\beta}{\beta+1}})^{-\beta-1}(\frac{1}{\varepsilon})^{pM},$$

for every $\varepsilon > 0$.

In the final part of this section, we mention that (in order to avoid very complicated computations) we did not pay attention to the fact that the best values of n_k and N that we obtained (namely $\frac{D\sqrt{M}}{\varepsilon_k^0}$ and $(\beta + 1)\frac{\ln(\frac{\varepsilon}{D\sqrt{M}})}{\ln(C)}$) may not be integers. In reality, we could work with $n_k = [\frac{D\sqrt{M}}{\varepsilon_k^0}] + 1$ and $N = [(\beta+1)\frac{\ln(\frac{\varepsilon}{D\sqrt{M}})}{\ln(C)}] + 1$ without a significant change.

5 Final remarks

Remark 5.1 We have:

$$\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}} \frac{\mathcal{C}_g(\varepsilon)}{\mathcal{C}_c(\varepsilon)} = \lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}} \frac{(1 - C^{\frac{\beta}{\beta+1}})^{-\beta-1}(\frac{1}{\varepsilon})^{pM}}{(x_0 L^{\frac{1}{p-1}})^{(\frac{1}{\varepsilon})^{\log_{\frac{1}{C}}(p)}}} = 0,$$

so, the grid algorithm is more efficient than the deterministic algorithm.

Remark 5.2 As $\left|u - [u + \frac{1}{2}]\right| \leq \frac{1}{2}$, we can improve our grid algorithm (which is based on the inequality $|u - [u]| < 1$) in the following way:

Pseudocode for the improved grid algorithm for GIFS

```
Read initially defined objects: constant: $L$, $M$, finite set, $m$ natural number: $K_0 \in P_{cp}([0, D]^M)$, mappings: $f_1, ..., f_L$, sequence: $(n_k)_k$, variables: $k$, $D_0$.
Initial values: $D_0 := K_0$.
        For $k$ from 1 to $m - 1$
            $D_1 := \{\frac{D}{n_k}[\frac{n_k}{D} f_l(u_1, ..., u_p) + \frac{1}{2}] \mid u_1, ..., u_p \in D_0, l \in \{1, ..., L\}\}$
            $D_0 := D_1$.
Print $D_m$.
```

Remark 5.3 On the one hand, repeating the arguments used in 4, A, for the case of an iterated function system (i.e., $p = 1$), we obtain that the complexity of the corresponding algorithm is described by the function $\mathcal{C} : (0, \infty) \to \mathbb{R}$ given by:

$$\mathcal{C}(\varepsilon) = (\frac{1}{\varepsilon})^{\frac{\ln L}{\ln \frac{1}{C}}},$$

for every $\varepsilon > 0$, so C is involved at the exponent of $\frac{1}{\varepsilon}$. We stress upon the fact that since $\lim\limits_{\substack{C \to 1 \\ C<1}} \frac{1}{\ln \frac{1}{C}} = \infty$, the closer is C to 1, the bigger is the number of points that we have to compute in order to approximate the attractor with an error less that ε.

On the other hand, in the rule that gives $\mathcal{C}_g(\varepsilon)$, the constant C is involved only in the coefficient $(1 - C^{\frac{\beta}{\beta+1}})^{-\beta-1}$.

Moreover, note that $\lim\limits_{\substack{C \to 1 \\ C<1}} \frac{\mathcal{C}_g(\varepsilon)}{\mathcal{C}_c(\varepsilon)} = 0$ for each $\varepsilon \in (0, 1)$.

6 Examples

In Section 4, we compared the algorithms with respect to a fixed preassigned error. In this section, our goal is to get an optimal image (with respect to the computer that we worked with) for three examples. For this reason, we chose a version of the grid algorithm for which $n_k = k^2$, the error being less than $D\sqrt{M}\left(\frac{1}{n^2} + C\frac{1}{(n-1)^2} + C^2\frac{1}{(n-2)^2} + ... + C^n\right)$, where n is the number of steps and C is the contraction constant of the system.

A.

Consider the GIFS $\mathcal{F} = (([0, 1]^2, d), \{f_1, f_2, f_3\})$, where, for $x = (x_1, y_1)$ and $y = (x_2, y_2)$, we have:

$$f_1(x, y) = (0.2x_1 + 0.2y_2; 0.2x_2 + 0.1y_2)$$

$$f_2(x, y) = (0.15x_1 + 0.07x_2 + 0.4; 0.15y_1 + 0.07y_2).$$

and

$$f_3(x, y) = (0.15y_1 + 0.07x_2; 0.15x_1 + 0.07y_2 + 0.04).$$

Using the deterministic algorithm, we get the image indicated in Fig. 1 and using the grid algorithm, we get the image in Fig. 2.

The deterministic algorithm ran 4 steps in 10 s, while the grid algorithm 8 steps in less than 10 s.

Figure 1 has approximately $3^{2^4} = 43,046,721$ points, while Fig. 2 comprises around 20,000 points.

Remark 6.1 If we allow the deterministic algorithm to run 5 steps, it needs about 90 min and we get a very similar image with the one in Fig. 2.

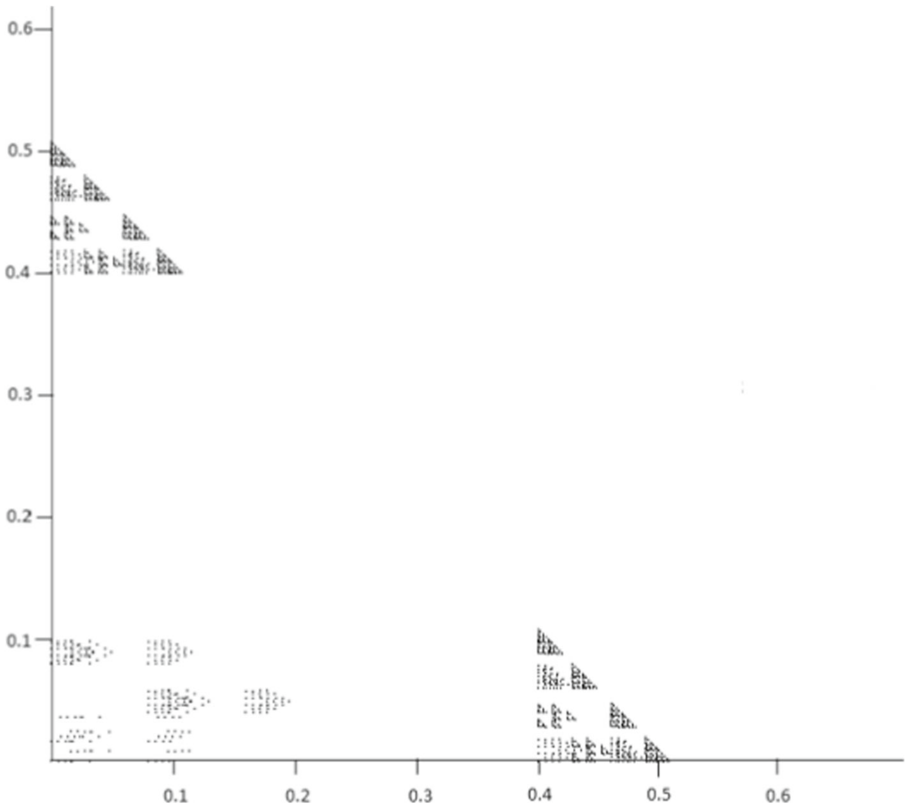

Fig. 1 Image obtained using the deterministic algorithm running 4 steps in 10 s

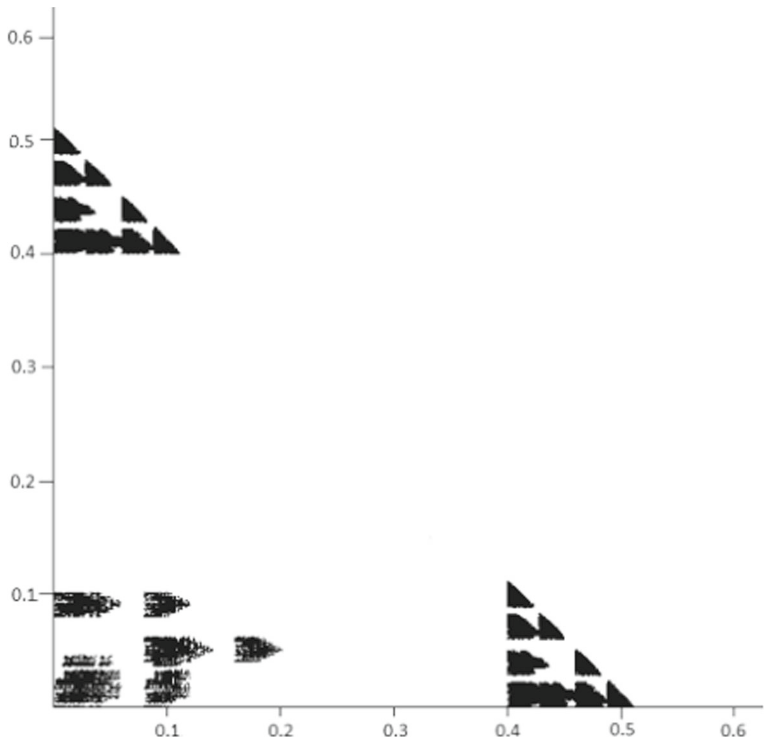

Fig. 2 Image obtained using the grid algorithm running 8 steps in less than 10 s

B.

Consider the GIFS $\mathcal{F} = (([0,1]^2, d), \{f_1, f_2\})$, where, for $x = (x_1, y_1)$ and $y = (x_2, y_2)$, we have:

$$f_1(x, y) = (0.1x_1 + 0.16y_1 - 0.01x_2 + 0.3y_2; -0.05y_1 + 0.15x_2 + 0.15y_2)$$

and

$$f_2(x, y) = (0.09x_1 - 0.1y_1 - 0.15x_2 + 0.14y_2 + 0.4; 0.14x_1 + 0.14y_1 + 0.14x_2 + 0.04).$$

The deterministic algorithm yields the image in Fig. 3 and the grid algorithm produces the image in Fig. 4.

The deterministic algorithm needed 20 s to run 5 steps, while the grid algorithm needed about 10 min to run 14 steps.

Figure 3 consists of about $2^{2^5} = 4,294,967,296$ points, while Fig. 4 is built up using around 2,000,000 points.

Remark 6.2 With the aid of the computer that we utilized, the deterministic algorithm would need 42 days to run 6 steps.

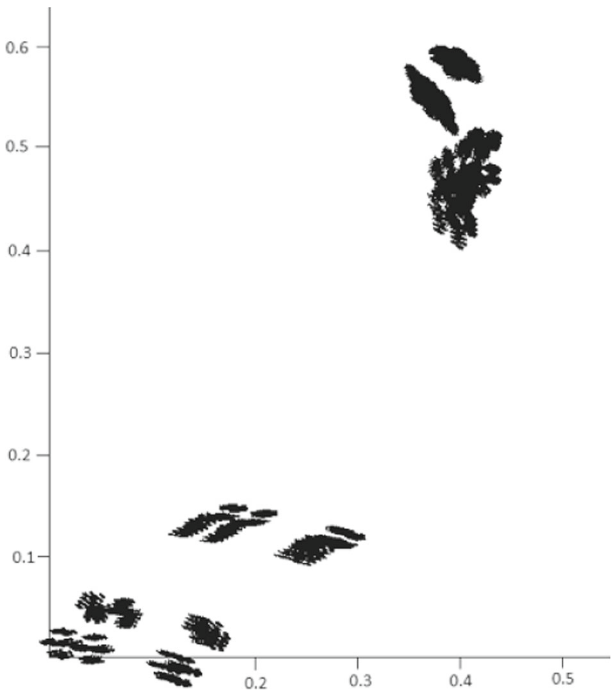

Fig. 3 Produced from using the deterministic algorithm which needed 20 s to run 5 steps

C.

Consider the GIFS $\mathcal{F} = (([0,1]^2, d), \{f_1, f_2\})$, where, for $x = (x_1, y_1)$ and $y = (x_2, y_2)$, we have:

$$f_1(x, y) = (0.5x_1 - 0.5y_1 + 0.001x_2 + 0.45; 0.5x_1 + 0.5y_1 + 0.001y_2 - 0.05)$$

and

$$f_2(x, y) = (0.2x_1 + 0.01x_2 + 0.14y_2 + 0.147; 0.2y_1 + 0.01y_2 + 0.105).$$

The image in Fig. 5 indicates what we get running the deterministic algorithm and the image in Fig. 6 what we obtain using the grid algorithm.

Both algorithms ran about 2 min, the deterministic one running 5 steps, while the grid one 22 steps.

Figure 5 consists of about $2^{2^5} = 4, 294, 967, 296$ points, while Fig. 6 is made up of circa 217,800 points.

Remark 6.3 Even though the number of points making up Fig. 5 is considerably bigger than the number of points building up Fig. 6, one can observe that the grid algorithm produces a much better approximation of the attractor.

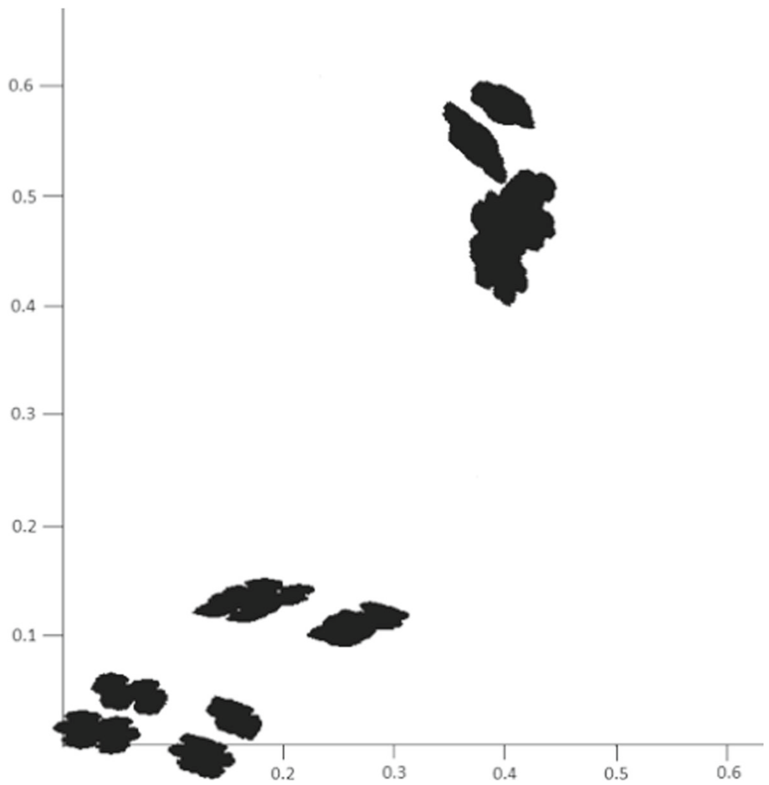

Fig. 4 Produced from using the grid algorithm which needed about 10 min to run 14 steps

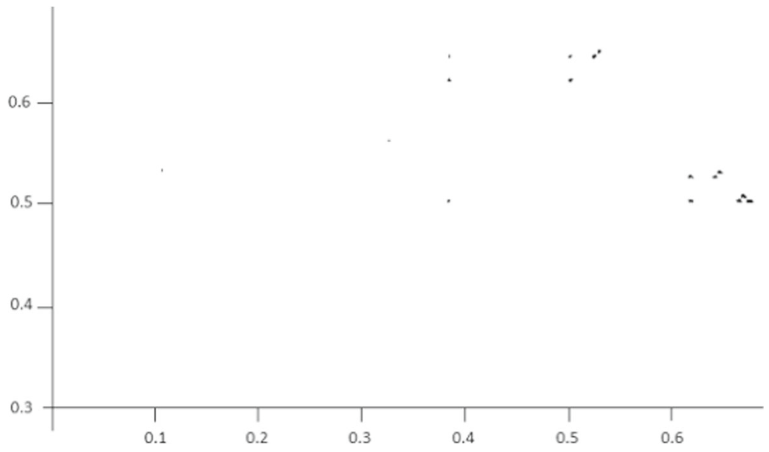

Fig. 5 The deterministic algorithm ran for about 2 min, running 5 steps

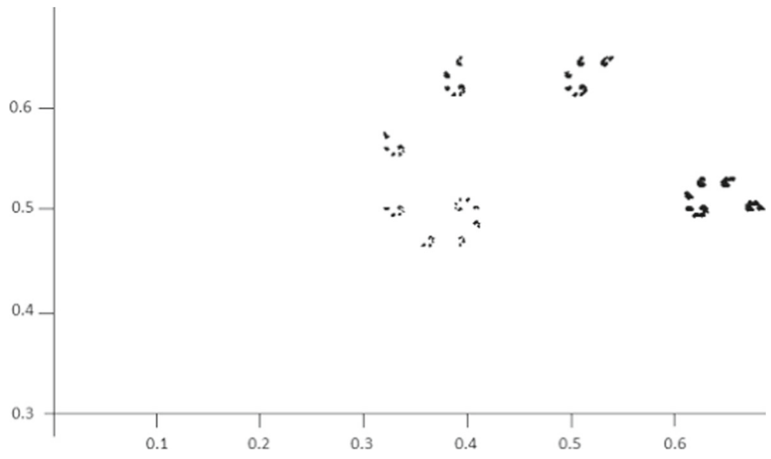

Fig. 6 The grid algorithm ran for about 2 min, running 22 steps

Acknowledgments The authors are very grateful to the reviewers whose extremely generous and valuable remarks and comments brought substantial improvements to the paper.

References

1. Browder, F.: On the convergence of successive approximations for nonlinear functional equations. Indag. Math. **30**, 27–35 (1968)
2. Hutchinson, J.E.: Fractals and self similarity. Indiana Univ. Math. J. **30**, 713–747 (1981)
3. Jaros, P., Maślanka, Ł., Strobin, F.: Algorithms generating images of attractors of generalized iterated function systems. Numer. Algorithm **73**, 477–499 (2016)
4. Maślanka, Ł., Strobin, F.: On generalized iterated function systems defined on l^∞-sum of a metric space. J. Math. Anal. Appl. **461**, 1795–1832 (2018)
5. Matkowski, J.: Integrable solutions of functional equations. Diss. Math. **127**, 68 (1975)
6. Miculescu, R.: Generalized iterated function systems with place dependent probabilities. Acta Appl. Math. **130**, 135–150 (2014)
7. Mihail, A., Miculescu, R.: Applications of Fixed Point Theorems in the Theory of Generalized IFS, Fixed Point Theory Appl. Volume 2008, Article ID 312876, 11 pages. https://doi.org/10.1155/2008/3128762017WR022284
8. Mihail, A., Miculescu, R.: A generalization of the Hutchinson measure. Mediterr. J. Math. **6**, 203–213 (2009)
9. Mihail, A., Miculescu, R.: Generalized IFSs on Noncompact Spaces, Fixed Point Theory Appl. Volume 2010, Article ID 584215, 11 pages. https://doi.org/10.1155/2010/584215
10. Oliveira, E.: The Ergodic Theorem for a new kind of attractor of a GIFS. Chaos Solitons Fractals **98**, 63–71 (2017)
11. Oliveira, E., Strobin, F.: Fuzzy attractors appearing from GIFZS. Fuzzy Sets Syst. **331**, 131–156 (2018)
12. Secelean, N.A.: Invariant measure associated with a generalized countable iterated function system. Mediterr. J. Math. **11**, 361–372 (2014)
13. Secelean, N.A.: Generalized iterated function systems on the space $l^\infty(x)$. J. Math. Anal. Appl. **410**, 847–858 (2014)
14. Strobin, F.: Attractors of generalized IFSs that are not attractors of IFSs. J. Math. Anal. Appl. **422**, 99–108 (2015)

15. Strobin, F., Swaczyna, J.: On a certain generalisation of the iterated function system. Bull. Aust. Math. Soc. **87**, 37–54 (2013)
16. Strobin, F., Swaczyna, J.: A code space for a generalized IFS. Fixed Point Theory **17**, 477–493 (2016)