IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 February 2023, accepted 1 March 2023, date of publication 6 March 2023, date of current version 17 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3253543

RESEARCH ARTICLE

Swarm and Evolutionary Algorithms in Image Compression by F-Transform

NGUYEN LE TOAN NHAT LINH[1] **AND QUOC BAO DIEP**[2]
[1]Applied Analysis Research Group, Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[2]Faculty of Mechanical-Electrical and Computer Engineering, School of Technology, Van Lang University, Ho Chi Minh City 700000, Vietnam
Corresponding author: Quoc Bao Diep (bao.dq@vlu.edu.vn)

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140).

ABSTRACT This article investigates the application of swarm and evolutionary algorithms, namely the SOMA, DE, and GA, for optimizing the F-transform-based image compression. To do this, we introduce the cost function, evaluating the approximation of decompressed images to the original image, concerning the parameters that control the approximation quality of the F-transform. This function is then minimized by the selected algorithms to find optimal settings for image compression and decompression. We design experiments to compare the performance of the original F-transform method and the methods optimized by SOMA, DE, and GA on a dataset of 10 pictures. In all considered cases, the results obtained with the optimized method completely surpass those obtained by the original one. We also apply a statistical test (called Wilcoxon signed-rank test) for ranking the performance of selected algorithms in this issue. The results show that the SOMA and DE perform well in cases where the compressed image sizes are small. However, the GA algorithm shows outperformance in comparison with the others in more complicated cases where the compressed image size is bigger. The outperformance of the GA is in terms of decompression quality and computation time. Finally, we provide a visual comparison between the original F-transform-based method and the method optimized by the GA, tested on a 128×128 picture. The decompressed image by the latter is much sharper and more detailed than that obtained by the former.

INDEX TERMS Image compression, swarm intelligence, evolutionary algorithms, numerical optimization.

I. INTRODUCTION

At the current time, the development of hardware devices has been faster than ever. The information and data have been processed faster and faster and saved in devices or networks with huge storage capacity. At a glance, it seems that the reduction of data size to enhance the processing speed as well as to save the memory is not so necessary. However, it is a fact that people always want things to be faster and storage devices, no matter how large, have limits. Therefore, data compression is still necessary despite the development of hardware devices. The image compression also follows the rule. It has become indispensable in real-world applications such as photography [1], remote sensing services [2], satellite image transmission [3], television broadcasting [4], and so on. There are numerous techniques introduced for image compression and F-transform-based image compression is one of them, getting a lot of attention recently.

A. HISTORICAL REVIEWS

The F-transform (fuzzy transform) was first introduced by Irina Perfilieva in [5] and then elaborated in [6]. Inspired by the Takagi-Sugeno fuzzy system [7] and classical transforms such as the Fourier and Laplace transforms, the F-transform consists in two phases (namely, the direct and inverse transforms), established with a fuzzy partition of a function domain (a set of fuzzy sets fulfilling the so-called à la Ruspini condition). In particular, the direct phase transforms a given function into a finite-dimensional vector whose components (called F-transform components) can be visualized as the

The associate editor coordinating the review of this manuscript and approving it for publication was Qingli Li.

weighted averages of the function on the regions covered by corresponding fuzzy sets. In inverse, the inverse phase approximately reconstructs the original function from the F-transform components, formed by the linear combination of the components and the corresponding fuzzy sets in the used fuzzy partition. Note that the approximation of a function by the F-transform is controlled by parameters of the fuzzy partition (among those, the denser fuzzy partition the better approximation). Moreover, by the computational simplicity, noise reduction ability, robustness, and ability to approximate functions from a few numbers of F-transform components, the F-transform has been successfully applied in many fields, including image processing.

The F-transform-based image compression was first introduced in [8] at the international workshop on fuzzy logic and applications (WILF) in 2005. The authors introduced the F-transform of binary discrete functions and provided a method for using it in image compression. In the compression phase, an image is converted to a smaller size based on its direct F-transform. More precisely, each pixel of the compressed image gets a value of the corresponding F-transform components. To reconstruct from the compressed image an image of the original size (decompression), the pixel values of the compressed image are combined with the corresponding fuzzy sets in a fuzzy partition to interpolate the pixel values of the reconstructed image. This is nothing else than the inverse F-transform of the original image. By the approximation of the inverse F-transform to the function that it is applied to, the decompressed image is close to the original one. The quality of the compression, which is assessed by the compression ratio and the closeness of the reconstructed image to the original, depends on the approximation quality of the F-transform.

B. MOTIVATION

Our works in this paper are motivated by previous investigations aimed at improving the quality of the F-transform-based image compression and the outperformance of bio-inspired optimization algorithms in recent competitions.

Since the first paper [8], there has been a lot of research to enhance the performance of the F-transform-based image compression. Most of these studies are technical treatments to retain as much information as possible during the compression process, for example, dividing an image into sub-blocks and applying the original F-transform method to them [9], [10], [11] or integrating the F-transform to the JPEG algorithm [12], [13]. Some studies are motivated by theoretical research to enhance the approximation ability of the F-transform, for example, the compression with the higher-degree F-transform [14] or with adjoint fuzzy partitions [1]. While few studies try to optimize key parameters which control the approximation by the F-transform. To the best of our knowledge, [15] is the only work in such a direction. The paper suggests optimizing the construction of fuzzy partitions by the gravitational search algorithm, an optimization algorithm based on Newton's law [16].

More popular and more efficient, swarm and evolutionary algorithms, bio-inspired algorithms instead of Newton's law (another class of optimization algorithms), are often used to solve complex optimization problems. They are inspired by the competition-cooperation of individuals in a population of creatures, such as finding food and protecting the nest (known as swarm intelligence (SI)) or imitating the genetics-mutation of the organism's genome (known as evolutionary algorithms (EAs)). The outperformance of swarm and evolutionary algorithms has been proven at IEEE CEC competitions. For example, a self-adaptive spherical search algorithm (SASS) [17] won the highest rank at the IEEE CEC Competitions on Real-World Single Objective Constrained Optimization or an improved multi-operator differential evolution algorithm (IMODE) [18] won the IEEE CEC Competition on Single Objective Bound Constrained Numerical Optimization, and so on.

The success and efficiency of bio-inspired optimization algorithms motivate us to conduct this study that is to apply swarm and evolutionary algorithms for improving the image compression by F-transform.

C. MAIN CONTRIBUTIONS

This paper investigates the application of bio-inspired optimization algorithms, namely, the genetic algorithm (GA) [19], the differential evolution (DE) [20], and the self-organizing migrating algorithm (SOMA) [21], to optimize the F-transform-based image compression. In particular, we introduce a class of fuzzy partitions determined with respect to parameters that will be considered as variables affecting the quality of the image compression. We introduce an objective (also known as cost or fitness) function to the problem with respect to those variables. The GA, DE, and SOMA algorithms are then applied to the function to find optimal solutions (parameters for constructing fuzzy partitions so that the compression achieves the highest quality). This research is illustrated with a dataset of ten selected pictures. The obtained results by the selected optimization algorithms are compared to each other and with the original F-transform-based image compression by statistical tests.

D. PAPER STRUCTURE

Section II is to introduce the principle of the selected swarm and evolutionary algorithms. In Section III, a class of fuzzy partitions defined with parameters is introduced and an objective function measuring the quality of the F-transform-based image compression is established in Section IV. Experimental tests are provided in Section V. The last, Section VI is to conclude.

II. SWARM AND EVOLUTIONARY ALGORITHMS

This section introduces bio-inspired optimization algorithms including swarm and evolutionary algorithms and explains why to apply SI and EAs algorithms.

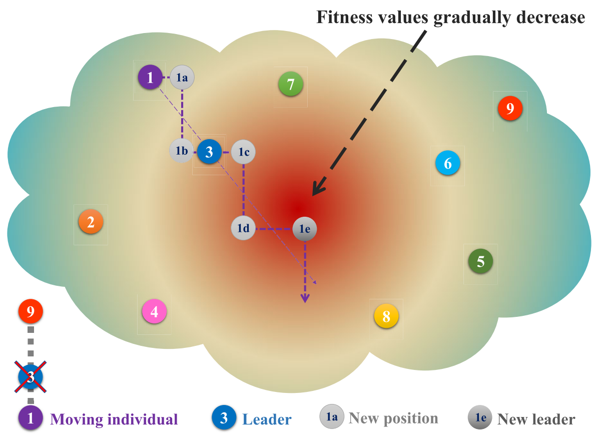


FIGURE 1. The movement mechanism of individuals in SOMA.

A. THE SOMA ALGORITHM

First published in the 2000s at an international conference in the Czech Republic, the self-organizing migrating algorithm (SOMA) is known as a bio-inspired optimization algorithm finding an optimal solution on continuous and bounded domains [21]. The authors observed the foraging process of organisms, i.e., when a food source is found, individuals share information with the rest of the population. On their way to the target, other individuals explore a better food source if available, they will share new information with others again, and the population will turn their attention to the new food source. Inspired by that process, the SOMA was introduced [21], [22], and [23].

1) INITIALIZATION METHOD

The algorithm starts by initializing a population containing random individuals that are possible solutions to the image compression problem, using (1). The population distribution is uniform over the entire search space (see Figure 1). Individuals are assigned scores that are their fitness function values. Then, strategies for organizing and migrating between them are implemented in the search space to explore and exploit promising sub-spaces with minimum fitness values under computer loops.

$$np_{population} = u_{min}^{j^{th}} + r\,(u_{min}^{j^{th}} - u_{max}^{j^{th}}) \tag{1}$$

where:

- $np_{population}$: the population with the size of NP,
- $u_{min}^{j^{th}}$: the minimum value of the j^{th}, $j = (1, 2, \ldots NP)$,
- $u_{max}^{j^{th}}$: the maximum value of the j^{th},
- r: the random number, $r \in [0\ 1]$.

2) ORGANIZATION METHOD

At the beginning of the loop, the population is organized into two separate parts: the leader with the highest score and the moving individuals remaining. They will one by one move toward the leader in the search space.

3) MIGRATION METHOD

That describes how an individual jumps toward the leader. In the SOMA algorithm, jumping is considered as the change of position on each dimension, gradually approaching the leader and moving away as described in (2). The jumping step is determined by a parameter named *Step*, and the jumping probability is determined by the parameter named *PRT*.

$$u_{ML+1}^{n,j} = u_{ML}^{c,j} + (u_{ML}^{l,j} - u_{ML}^{c,j})\ k\ PRTVector_j \tag{2}$$

where:

- $u_{ML+1}^{n,j}$: the new position created for the next iteration loop,
- $u_{ML}^{c,j}$: the selected individual used for moving,
- $u_{ML}^{l,j}$: the leader individual used for moving,
- k: the discrete step of moving, $Step : Step : PathLength$,
- $if\ r_j < PRT;\ PRTVector_j = 1;\ \ else,\ 0.$

4) EVALUATION METHOD

During migration, the generated position that has a lower fitness value than the original can be discovered and retained. The old will be eliminated to preserve the population size.

Those processes occur in Migration loops until the algorithm reaches the specified stopping condition. The SOMA algorithm is pseudocoded as in **Algotithm 1**.

Algorithm 1 Self-organizing migrating algorithm

```
input  : SOMA control parameters
output: RMSE, Xnode, Ynode
$np_{population} \leftarrow \{u \mid u_{min}^{j^{th}} \le u \le u_{max}^{j^{th}},\ j = (1, 2, \ldots NP)\}$ ;
$f_{u_{pop}} \leftarrow f_{(u_{pop})}$ ;
while $FEs \le MaxFEs$ do
    $u_{leader} \leftarrow \{u \mid \min(f_{u_{pop}})\}$ ;
    for $i \leftarrow 1$ to $np_{population} \setminus leader$ do
        $u_{moving} \leftarrow np_{population}^{j^{th}}$ ;
        $u_{path} \leftarrow u_{moving}$ move to $u_{leader}$ ;
        $f_{u_{path}} \leftarrow f_{(u_{path})}$ ;
        $np_{population}^{j^{th}} \leftarrow \{u \mid \min(f_{u_{moving}}, f_{u_{path}})\}$ ;
    end
end
```

Over the past two decades, SOMA has evolved in two main aspects, one is to increase the performance of the algorithm, and the other is to solve different classes of problems.

The performance-enhanced versions focus on two main streams. The first is to improve the algorithm itself by suggesting better control parameters such as SOMA with clustering-aided migration [24] and leader selection in SOMA [25], and reorganizing the mechanism the algorithm operates such as self-adapting SOMA [26] and the ensemble of strategies and perturbation parameter in SOMA [27]. The second is to combine SOMA with other algorithms such as GA and PSO to form hybridization versions like [28] and [29], please refer to [30] for more details.

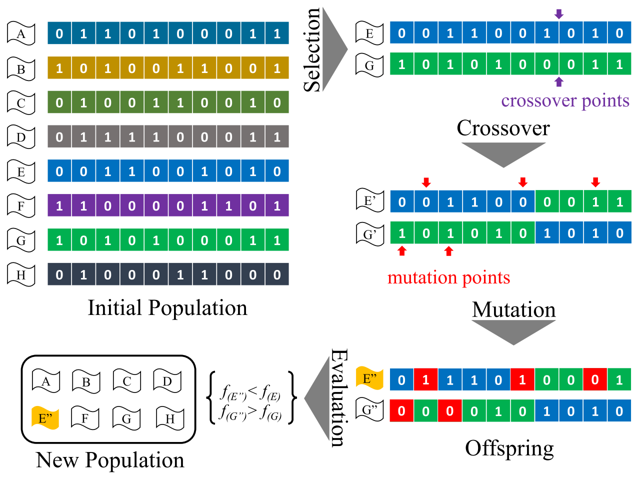


FIGURE 2. The principle of GA.

Regardless of the stream, SOMA has obtained certain achievements, most recently being tied with the HyDE algorithm in the top 4 strongest algorithms in the well-known CEC2019 competition [31].

B. THE GENETIC ALGORITHM

First introduced in 1989, the Genetic Algorithm (GA) is a method for solving optimization problems that imitate the inheritance and variational processes of an organism's genome [19]. Differing from swarm intelligence, which exchanges information between individuals in a population within the same generation, genes in GA inherit biological characteristics from generation to generation to produce offspring from their parents for the next generation.

In particular, an important process applied will mutate some points in the offspring genes, making them appear as novel features that have never been seen in the population before, as shown in Figure 2. Affected by the mechanism of natural selection (cost function in this case), individuals carrying bad characteristics after mutation will be eliminated and good individuals will be kept and used for further generations. The GA can be described as follows.

Initialization: The random feasible solutions of the optimization problem will be encoded according to a certain rule to form chromosomes, gathering into an initial population. They can be real numbers, integers, ordinal numbers, etc. All individuals are then evaluated by the defined cost function.

Selection: The chromosomes are selected randomly or according to a certain strategy that is usually based on fitness value, known as the parents. They will be used for the next step.

Crossover: It takes place after the selection, in order to swap part of the parental chromosomes with each other to create offspring chromosomes that combine the genetic characteristics of the parents. In GA, many different crossover methods are implemented depending on the specific problem.

Mutation: A random or intentional change in one or more positions on offspring chromosomes to produce other novel chromosomes that have never existed in a population before.

Evaluation: The offspring chromosomes will be evaluated by the defined cost function to remove the bad mutant chromosomes and keep the good ones.

The mentioned activities are repeated in computer loops till the GA satisfies the given stopping condition. The GA algorithm is pseudocoded as in **Algotithm 2**.

Algorithm 2 Genetic Algorithm

```
input  : GA control parameters
output: RMSE, Xnode, Ynode
[Initialization] $np_{chr} \leftarrow \{chromo_i \overset{encode}{:=} c,$
$c^{j^{th}}_{min} \leq c \leq c^{j^{th}}_{max}, j = (1, 2, \ldots NP)\}$ ;
[Evaluation] $f_{np_{chr}} \leftarrow f_{(np_{chr})}$;
while $FEs \leq MaxFEs$ do
    [Selection]
    $chr_{parent} \leftarrow \{chr_j \subset np_{chr}, j = (1, 2, \ldots NP)\}$ ;
    [Crossover]
    $(chromo_I, chromo_{II}) \leftarrow C_{(chr_{parent})}$ ;
    [Mutation]
    $(chromo_x, chromo_y) \leftarrow M_{(chromo_I, chromo_{II})}$ ;
    [Evaluation]
    $f_{chromo_{xy}} \leftarrow f_{(chromo_{(chromo_x, chromo_y)})}$ ;
    [Accepting]
    $chr_j \leftarrow \{chr \mid \min(f_{chromo_{xy}}, f_{chr_{parent}})\}$ ;
end
```

Since its introduction, the GA has been developed into many different versions and is widely used in real-world applications. Its development directions can be divided into four main categories as listed below and detailed in [32].

- Hybrid genetic algorithms are combined with other algorithms to generate versions of so-called Hybrid GAs to increase solutions quality, and better performance, such as a hybrid genetic algorithm and tabu search [33] and hybrid genetic firefly algorithm-based [34].
- Parallel genetic algorithms aim to improve the computation time and performance of the GA, such as [35] and [36].
- Chaotic genetic algorithms aim to preserve the diversity of the population and avoid premature convergence of the algorithm, such as [37] and [38].
- Encoding-based-method genetic algorithms include real and binary chromosomes depending on the problem being solved, such as [39] and [40].

C. THE DE ALGORITHM

Published in 1997, Differential Evolution (DE) was known as a nature-inspired metaheuristic for solving numerical optimization problems [20]. Over the past two decades, DE with many powerful and robust variants has emerged and beat other algorithms in the IEEE CEC competitions.

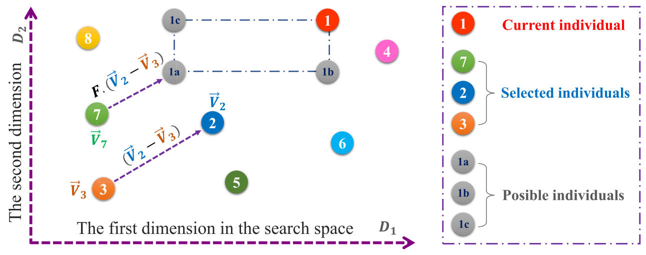


FIGURE 3. The principle of DE.

The backbone of DE mimics the organism-natural evolution and goes through similar steps as many other evolutionary algorithms which are briefly described as follows, and briefly summarized in Figure 3.

Initialization: DE generates an initial population containing NP individuals uniformly distributed over the whole searching range, as in (3). All individuals are then evaluated by the defined cost function, and the algorithm enters the first iteration.

$$\vec{X}_{i,gen} = [x_{1,i,gen},\ x_{2,i,gen},\ \ldots,\ x_{D,i,gen}], \tag{3}$$

where $x_{j,i,gen} = x_{j,lo} + r\ (x_{j,up} - x_{j,lo})$.

Selection: For each individual (known as a target vector) in the population till the end, the algorithm randomly selects three other separate individuals to create the new one as in the next step.

Mutation: By applying (4), the DE creates a new individual (known as a donor vector) from the selected above.

$$\vec{V}_{i,gen} = \vec{X}_{r1,gen} + F.(\vec{X}_{r2,gen} - \vec{X}_{r3,gen}) \tag{4}$$

Crossover: This process makes dimensional-swapping between the individual just created and the initial to obtain a new individual (known as a trial vector), which has characteristics never been seen before (see (5)).

$$u_{j,i,gen} = \begin{cases} v_{j,i,gen} & \textit{if } (r\ \le Cr) \textit{ or } j = j_r \\ x_{j,i,gen} & \textit{otherwise} \end{cases} \tag{5}$$

Evaluation: The new individual is then evaluated and put back into the population if it has a better cost value than the selected individual.

The DE activities are repeated in computer loops till the algorithm meets the given stopping condition. Figure 3 depicts the DE algorithm. The DE algorithm is pseudocoded as in **Algotithm 3**.

DE has achieved great success by virtually dominating all CEC competitions, for example, the jDE100 won the CEC 2019 [41], the improved multi-operator differential evolution algorithm (IMODE) won the CEC 2020 [18], NL-SHADE-RSP ranked the position in the overall performance of CEC 2021 [42], refers to [43] for more details.

The development and success of the above algorithms are summarized in publications [30], [32], and [43]. In-depth readers can refer to them for more details.

Algorithm 3 Differential Evolution, DE/rand/1/bin

```
input : DE control parameters
output: RMSE, Xnode, Ynode
[Initialization] $np \leftarrow \{\vec{X}_i\ i = (1, 2, \ldots NP)\}$ ;
[Evaluation] $f_{np} \leftarrow f_{(np)}$ ;
while $FEs \le MaxFEs$ do
    for $a \leftarrow 1$ to $NP$ do
        [Selection]
        $(\vec{X}_{n1}, \vec{X}_{n2}, \vec{X}_{n3}) \leftarrow \vec{X}_n \subset np, n = [1 \ldots NP]$ ;
        [Mutation]
        $\vec{V}_a \leftarrow \vec{X}_{n1} + F.(\vec{X}_{n2} - \vec{X}_{n3})$ ;
        [Crossover]
        $\vec{U}_a \leftarrow \{u_{da}|u_{da} = v_{da} \leftrightarrow (rand_{da} \le Cr|$
        $d = d_{rand})|u_{da} = x_{da}, v_{da} \in \vec{V}_a,$
        $x_{da} \in \vec{X}_a, d = (1, 2, \ldots Dimension)\}$ ;
        [Evaluation]
        $f_{\vec{U}_a} \leftarrow f_{(\vec{U}_a)}$ ;
        [Accepting]
        $\vec{X}_a \leftarrow \{\vec{X}_a \mid \vec{U}_a \mid \min(f_{\vec{X}_a}, f_{\vec{U}_a})\}$ ;
    end
end
```

III. F-TRANSFORM-BASED IMAGE COMPRESSION

This section provides abstract knowledge about the binary discrete F-transform and the framework for applying it to image compression.

A. THE F-TRANSFORM OF BINARY DISCRETE FUNCTIONS

The F-transform of binary discrete functions was first proposed by I. Perfilieva in [8]. The central notion of this transform is the fuzzy partitions of rectangles.

Definition 1: Let $[a, b] \times [c, d]$ *be a rectangle in* $\mathbb{R}^2$, *and two integers* M, N *be such that* $M, N \ge 3$. *A family* $\{\mathcal{K}_{k\ell} : [a, b] \times [c, d] \to [0, 1] \mid k = \overline{1, M}, \ell = \overline{1, N}\}$, *is said to be a fuzzy partition of* $[a, b] \times [c, d]$, *if*

$$\sum_{k=1}^{M} \sum_{\ell=1}^{N} \mathcal{K}_{k\ell}(t, s) = 1,$$

for any $(t, s) \in [a, b] \times [c, d]$. *In this statement,* $\mathcal{K}_{k\ell}$ *is called the* $k\ell$*-th basic function of the fuzzy partition.*

The F-transform of a binary discrete function consists of two phases, namely the direct and inverse F-transform.

Definition 2: Let $D = \{(i, j) \mid i = \overline{1, m},\ j = \overline{1, n}\}$, *and let* $I : D \to \mathbb{R}$ *be a discrete function. Let* $\mathcal{A} = \{\mathcal{K}_{k\ell} \mid k = \overline{1, M}, \ell = \overline{1, N}\}$ *be a fuzzy partition of* $[1, m] \times [1, n]$. *The direct F-transform of* I *with respect to* $\mathcal{A}$ *is the matrix, denoted by* $F[I]$ *and given as follows*:

$$F[I] = \begin{bmatrix} F_{11}[I] & F_{12}[I] & \ldots & F_{1M}[I] \\ F_{21}[I] & F_{22}[I] & \ldots & F_{2M}[I] \\ \vdots & \vdots & \ldots & \vdots \\ F_{N1}[I] & F_{N2}[I] & \ldots & F_{NM}[I] \end{bmatrix}, \tag{6}$$

with

$$F_{k\ell}[I] = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} I(i,j)\mathcal{K}_{k\ell}(i,j)}{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathcal{K}_{k\ell}(i,j)}, \tag{7}$$

for $k = \overline{1,M}$, $\ell = \overline{1,N}$. *The component* $F_{k\ell}[I]$ *is called the* $k\ell$*-th F-transform components.*
Note that (6) is well-defined if, for any $k = \overline{1,M}$, $\ell = \overline{1,N}$, there exists $(i,j) \in D$ such that $\mathcal{K}_{k\ell}(i,j) > 0$. This condition is known as the density condition of fuzzy partition $\mathcal{A}$ on D.

Definition 3: Let D, I *and* $\mathcal{A}$ *be given as in Definition 2, and let* $F[I]$ *be the direct F-transform of* I *with respect to* $\mathcal{A}$ *as in (6). The inverse F-transform of* I *with respect to* $\mathcal{A}$ *is a discrete function denoted by* $\tilde{I}$ *and defined by*

$$\tilde{I}(i,j) = \sum_{k=1}^{M}\sum_{\ell=1}^{N} F_{k\ell}[I]\mathcal{K}_{k\ell}(i,j), \quad (i,j) \in D.$$

The inverse function $\tilde{I}$ provides an approximation to the original discrete function I (see [6]). The precision of this approximation is driven by the way how the fuzzy partition is constructed, in particular, the denser fuzzy partitions[1] the better approximations.

B. IMAGE COMPRESSION BY THE F-TRANSFORM

This subsection describes in an abstract frame of the image compression by the F-transform initiated by [8].

Let $D = \{(i,j) \mid i = \overline{1,m},\ j = \overline{1,n}\}$, and let $I : D \to [0,1]$ be a grayscale image. With the F-transform, we can convert I to a smaller size image, and then, reconstruct from it a full-size image as the original. More precisely, to compress I to an image C of size $M \times N$, $M < m$, $N < n$, by the F-transform, we follow the following steps:

- S1: set up a fuzzy partition $\{\mathcal{K}_{k\ell} \mid k = \overline{1,M}, \ell = \overline{1,N}\}$ of $[1,m] \times [1,n]$,
- S2: compute the F-transform components $F_{k\ell}[I]$, for $k = \overline{1,M}$, $\ell = \overline{1,N}$, by (7),
- S3: construct image $C : \{1,\ldots,M\} \times \{1,\ldots,N\} \to [0,1]$ from the direct F-transform $F[I]$ in (6) by $C(k,\ell) = F_{k\ell}[I]$, for any $k = \overline{1,M}$, $\ell = \overline{1,N}$.

To reconstruct a full size image $\tilde{I} : D \to [0,1]$ from C, we used the inverse F-transform function where the $k\ell$-th F-transform components is replaced by $C(k,\ell)$, namely,

$$\tilde{I}(i,j) = \sum_{k=1}^{M}\sum_{\ell=1}^{N} C(k,\ell)\mathcal{K}_{k\ell}(i,j), \quad (i,j) \in D.$$

IV. OPTIMIZE THE F-TRANSFORM-BASED IMAGE COMPRESSION

The key of the F-transform-based image compression is the fuzzy partition of rectangles (see Section III). In what follows, we introduce a particular class of fuzzy partitions of rectangles defined with respect to variable parameters that can be selected so that the compression reaches its best quality.

[1]Fuzzy partitions with a higher number of basic functions.

A. VARIABLE PARAMETERIZED FUZZY PARTITION OF RECTANGLES

Definition 4: A function $\phi : \mathbb{R} \to \mathbb{R}$ *is said to be a* fuzzy partition-generating function *(or* Fp-generating function, for short*) provided that it is continuous, increasing on* $[-1,0]$ *with* $f(-1) = 0$ *and* $\phi(0) = 1$.

Example 1:
1) *Linear Fp-generating function* $L(x) = x + 1$.
2) *Quadratic Fp-generating function* $Q(x) = 1 - x^2$.
3) *Sine Fp-generating function* $S(x) = \sin\left(\frac{\pi}{2}(x+1)\right)$.

Proposition 1: Let $[a,b] \times [c,d]$ *be a rectangle in* $\mathbb{R}^2$, *and let* ϕ *and* ψ *be Fp-generating functions. Let* t_k, $k = \overline{1,M}$, *and* s_ℓ, $\ell = \overline{1,N}$, *be nodes in* $[a,b]$ *and* $[c,d]$ *such that* $a = t_1 < \cdots < t_M = b$ *and* $c = s_1 < \cdots < s_N = d$, *respectively. Let* A_k, $k = \overline{1,M}$, *and* B_ℓ, $\ell = \overline{1,N}$, *be functions defined, respectively, on* $[a,b]$ *and* $[c,d]$ *as follows*:

$$A_1(t) = 1 - \phi\left(\frac{t - t_2}{t_2 - t_1}\right), \quad B_1(s) = 1 - \psi\left(\frac{s - s_2}{s_2 - s_1}\right)$$
$$A_M(t) = \phi\left(\frac{t - t_M}{t_M - t_{N-1}}\right), \quad B_N(s) = \psi\left(\frac{s - s_N}{s_N - s_{N-1}}\right),$$

where $A_1(t) = 0$ *if* $t > t_2$, $A_M(t) = 0$ *if* $t < t_{M-1}$, $B_1(s) = 0$ *if* $s > s_2$, $B_N(s) = 0$ *if* $s < s_{N-1}$, *and*

$$A_k(t) = \begin{cases} 1 - \phi\left(\dfrac{t - t_{k+1}}{t_{k+1} - t_k}\right), & t \in (t_k, t_{k+1}] \\ \phi\left(\dfrac{t - t_k}{t_k - t_{k-1}}\right), & t \in [t_{k-1}, t_k] \\ 0, & \textit{otherwise}, \end{cases}$$

$$B_\ell(s) = \begin{cases} 1 - \psi\left(\dfrac{s - s_{\ell+1}}{s_{\ell+1} - s_\ell}\right), & s \in (s_\ell, s_{\ell+1}] \\ \psi\left(\dfrac{s - s_\ell}{s_\ell - s_{\ell-1}}\right), & s \in [s_{\ell-1}, s_\ell] \\ 0, & \textit{otherwise}, \end{cases}$$

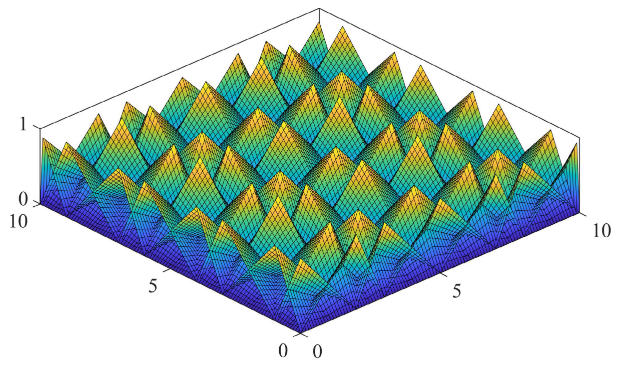
for $k = 2,\ldots,M-1$, $\ell = 2,\ldots,N-1$. *Then,* $\{\mathcal{K}_{k\ell}(t,s) = A_k(t)\cdot B_\ell(s) \mid k = \overline{1,M}, \ell = \overline{1,N}\}$ *forms a fuzzy partition of* $[a,b] \times [c,d]$.

Proof: Let $(t,s) \in [a,b] \times [c,d]$ be arbitrary. It is easy to see that $A_k(t), B_\ell(s) \in [0,1]$, for any $k = \overline{1,M}$ and $\ell = \overline{1,N}$. Assume that $(t,s) \in [t_k, t_{k+1}] \times [s_\ell, s_{\ell+1}]$, with $k = \overline{1,M-1}$, $\ell = \overline{1,N-1}$. Then, we have

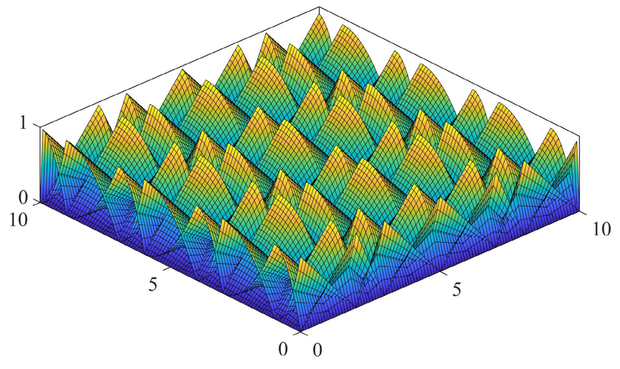
$$\sum_{k=1}^{M} A_k(t) = A_k(t) + A_{k+1}(t)$$
$$= 1 - \phi\left(\frac{t - t_{k+1}}{t_{k+1} - t_k}\right) + \phi\left(\frac{t - t_{k+1}}{t_{k+1} - t_k}\right) = 1$$

and

$$\sum_{\ell=1}^{N} B_\ell(s) = B_\ell(s) + B_{\ell+1}(s)$$
$$= 1 - \psi\left(\frac{s - s_{\ell+1}}{s_{\ell+1} - s_\ell}\right) + \psi\left(\frac{s - s_{\ell+1}}{s_{\ell+1} - s_\ell}\right) = 1.$$



(a) With two linear Fp-generating functions



(b) With the linear and quadratic Fp-generating functions

FIGURE 4. Variable parameterized fuzzy partitions of [0, 10] × [0, 10].

It follows that

$$\sum_{k=1}^{M}\sum_{\ell=1}^{N}\mathcal{K}_{k\ell}(t,s)=\sum_{k=1}^{M}\sum_{\ell=1}^{N}A_k(t)\cdot B_\ell(s)$$
$$=\sum_{k=1}^{M}A_k(t)\cdot\sum_{\ell=1}^{N}B_\ell(s)=1.$$

This means that $\{\mathcal{K}_{k\ell}\mid k=\overline{1,M},\ell=\overline{1,N}\}$ forms a fuzzy partition of $[a,b]\times[c,d]$. The proof is finished. $\square$

Definition 5: Let $[a,b]\times[c,d]$ *be a rectangle in* $\mathbb{R}^2$, ϕ *and* ψ *be Fp-generating functions, and nodes* t_k, $k=\overline{1,M}$, *and* s_ℓ, $\ell=\overline{1,N}$, *with* $M,N\geq 3$, *be such that* $a=t_1<\cdots<t_M=b$ *and* $c=s_1<\cdots<s_N=d$. *The fuzzy partition of* $[a,b]\times[c,d]$ *defined as in Proposition 1 is called the* variable parameterized fuzzy partition *determined with respect to nodes* t_k, $k=\overline{1,M}$, *and* s_ℓ, $\ell=\overline{1,N}$ *and Fp-generating functions* ϕ *and* ψ.

Example 2: Consider the rectangle $[0,10]\times[0,10]$ *and select nodes* $(0,2,3,5,7,8,10)$ *and* $(0,1,3,4,6,7,9,10)$. *In Figure 4, we display the variable parameterized fuzzy partitions of the rectangle determined with respect to selected nodes and linear and quadratic Fp-generating functions.*

B. OPTIMIZE THE IMAGE COMPRESSION

As in Subsection III-B, the F-transform can be used to compress an image to a smaller size and then reconstruct the original image from it. The quality of the reconstruction (how the reconstructed image is close to the original one) depends on the approximation quality of the F-transform driven by the settings of fuzzy partitions [8]. This subsection aims to provide methods based on swarm and evolutionary algorithms to establish fuzzy partitions leading to optimal results in image compression.

Let I be a grayscale image of size $m\times n$, represented as follows:

$$I:\{1,\ldots,m\}\times\{1,\ldots,n\}\to[0,1].$$

Assume that the goal is to compress I to an image C of size $M\times N$, $3\leq M<m$, $3\leq N<n$ and then reconstruct from C an image of the original size $m\times n$. By the F-transform-based method, this issue is solved according to the procedure mentioned in Subsection III-B. To be able to optimize this procedure, we propose to use the F-transform with respect to variable parameterized fuzzy partitions determined with nodes and Fp-generating functions. Namely, the fuzzy partition in step S1 in Subsection III-B is established according to the following steps:

S1a: choose Fp-generating functions ϕ and ψ,
S1b: select nodes t_k, $k=\overline{1,M}$, and s_ℓ, $\ell=\overline{1,N}$, satisfying that $1=t_1<\cdots<t_M=m$ and $1=s_1<\cdots<s_N=n$.
S1c: construct a variable parameterized fuzzy partition of $[1,m]\times[1,n]$ with respect to selected nodes and two Fp-generating functions.

By this approach, the compressed image as well as the reconstructed one are driven by selected nodes and Fp-generating functions. Assume that the latter are fixed (for the computation in the next section, we fix to use the linear Fp-generating function), we proposed to use the swarm and evolutionary algorithms introduced in Section II to select nodes t_k, $k=\overline{1,M}$, and s_ℓ, $\ell=\overline{1,N}$, that maximize the quality of the reconstructed image.

Let $C[t_1,\ldots,t_M,s_1,\ldots,s_N]$ and $\tilde{I}[t_1,\ldots,t_M,s_1,\ldots,s_N]$ be the compressed and reconstructed images from I obtained by the F-transform method specified to the variable parameterized fuzzy partition established in the previous steps with respect to nodes t_k, $k=\overline{1,M}$, and s_ℓ, $\ell=\overline{1,N}$. To evaluate the quality of the reconstructed image, we use the Root Mean Square Error (RMSE) defined by

$$RMSE(I,\tilde{I}[t_1,\ldots,t_M,s_1,\ldots,s_N])$$
$$=\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(I(i,j)-\tilde{I}[t_1,\ldots,t_M,s_1,\ldots,s_N](i,j))^2}{mn}}.$$

Let

$$f(t_1,\ldots,t_M,s_1,\ldots,s_N)$$
$$=\mathrm{RMSE}(I,\tilde{I}[t_1,\ldots,t_M,s_1,\ldots,s_N])$$

and consider it as an objective function of variables $(t_1,\ldots,t_M,s_1,\ldots,s_N)$. The optimization algorithms are applied to find solutions that minimize the function under the

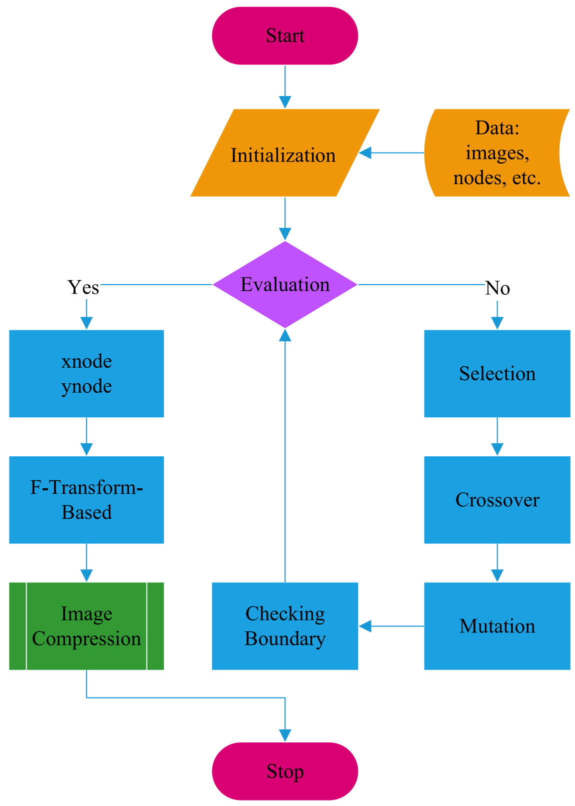


FIGURE 5. The optimized F-transform-based image compression flowchart.

constraints that

$$1 = t_1 < t_2 < \ldots < t_{M-1} < t_M = m,$$
$$1 = s_1 < s_2 < \ldots < s_{N-1} < t_N = n$$

and the searching ranges of variables $t_1, \ldots, t_M$ and $s_1, \ldots, s_N$ are the sets $\{1, \ldots, m\}$ and $\{1, \ldots, n\}$, respectively.[2] Figure 5 describes the whole F-transform-based image compression optimized by the swarm and evolutionary algorithms.

V. ILLUSTRATIONS

This section is to analyze the application of the SOMA, DE, and GA algorithms in image compression with the F-transform. This includes comparisons between the original F-transform-based method and the methods optimized by the mentioned algorithms.

A. NUMERICAL COMPARISION

We select a dataset of 10 images as in Figure 6, taken by the authors. Each image in the dataset is in the grayscale with a unique size 16 × 16. The original F-transform-based image compression (FT) and the methods optimized by the

[2]In general, the search ranges can be intervals $[1, m]$ and $[1, n]$. However, the discrete F-transform only depends on the number of discrete points covered by each basic function in the used fuzzy partition (see [6]). Therefore, it is sufficient to search in the sets $\{1, \ldots, m\}$ and $\{1, \ldots, n\}$.

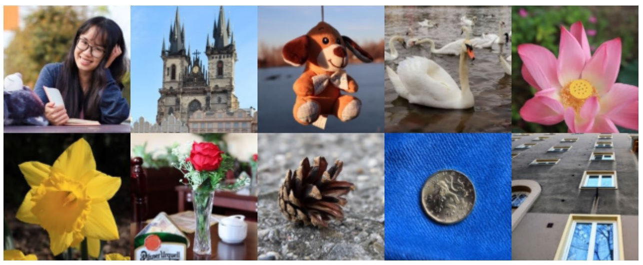


FIGURE 6. The image dataset, displayed in a resolution of 128 × 128 pixels.

TABLE 1. The control parameter values of three selected swarm and evolutionary algorithms.

Algorithms	The setting values used for 25 runs
SOMA	$PopSize = 100; PathLength = 2.0; PRT = 0.08; Step = 0.11; MaxFEs = 10^5$
DE	$PopSize = 100; Cr = 0.3; F = 0.7; MaxFEs = 10^5$
GA	$PopSize = 100; InitialPopulationRange = [0; imagesize]; MaxGens = 1000$

SOMA, DE, and GA (FT-SOMA, FT-DE, and FT-GA) are applied in turn to each image to compress and reconstruct it. Particularly, the compression is performed with three ratios corresponding to three compressed image sizes, 4 × 4, 8 × 8, and 12 × 12. In this experiment, the FT method is performed with uniform fuzzy partitions constructed from nodes that are uniformly selected along two sides of rectangles (see [8]). On the other hand, since the SOMA, DE, and GA algorithms are implemented with random components that randomly affect the obtained results, the experiment with FT-SOMA, FT-DE, and FT-GA methods is repeated 25 times for each image.

Our tests are performed on the Karolina Supercomputer in the Czech Republic,[3] using CPUs of 4 × 2x AMD 7452, 256 cores in total (two compute nodes), 2.35 GHz, and 512 GB RAM. The operating system used is Centos 64 bit 7.x, with MATLAB R2022a version.

The control parameters of the optimization algorithms are set by default according to the recommendations in the original publications [19], [20], [23] and specified in Table 1. For GA, the *Global Optimization Toolbox* is used with the default setting of MATLAB, except for some other parameters pointed out in Table 1.

We first compare the four mentioned methods according to the approximation of reconstructed images to the original ones, measured by The Root Mean Square Error (RMSE). Note that the lower error the better compression. The obtained results are given in Tables 2, 3 and 4. In these tables, for the optimized methods, we evaluate the mean and standard deviation of the RMSE obtained from 25 independent runs. It is clear that the original FT method is defeated by all optimized methods over all cases and all images in the dataset.

[3]https://www.it4i.cz/en/infrastructure/karolina

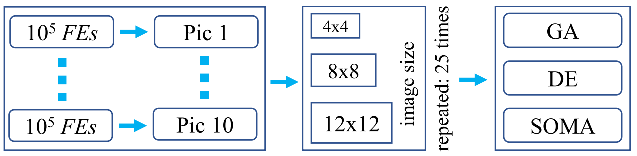


FIGURE 7. The design of experiments for optimization algorithm performance testing.

Particularly, the errors obtained by the FT method are significantly higher than the mean error of the other methods. This shows that the application of optimization algorithms, namely SOMA, DE, or GA, has significantly improved the quality of the image compression based on F-transform. On another side, we used the Wilcoxon signed-rank test (WRT)[4] to compare the performance of FT-SOMA, FT-DE, and FT-GA methods so that the comparison is statistically significant. Particularly, we use the function `signrank`(x, y) in MATLAB with default settings (5% significant level) to order the performance of each pair of optimized methods based on the RMSE from 25 runs. Note that when two methods are incomparable (the p-value is higher than the significance level), we put them in the same rank. The obtained ranking is also shown in Tables 2, 3, 4 (between the bracket). Note that the worst of the FT method is clear, so we do not sign its rank. Especially, in Table 2, since all optimized methods provide the same results, we do not compare them. It follows from the ranking that the application of different optimization algorithms makes a difference only when the complexity of the cost function is increased with more variables. In particular, FT-SOMA, FT-DE, and FT-GA show the difference in cases where the compressed image sizes are 8×8 and 12×12. Especially, in the latter case (the most complicated case where the objective function is of 20 variables), the GA shows the best performance among the selected algorithms.

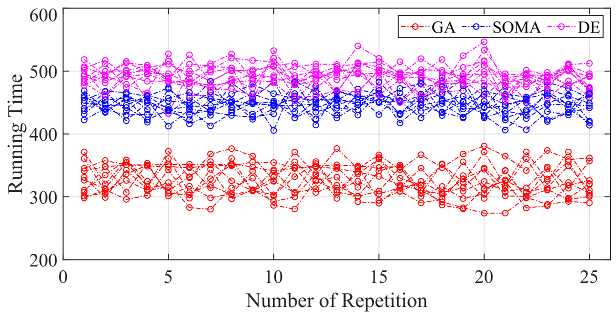
On the other hand, we also compare the performance of FT-SOMA, FT-DE, and FT-GA with respect to the computation time. To do this, we measure the length of time corresponding to each method that needs to complete the computational process on each picture once (including the search of optimal nodes, compression, and reconstruction). The obtained results a displayed in Figure 8. One can see that the GA performed faster than the other algorithms in the most complicated case (the compressed image size is 12×12). As mentioned above, in this case, the objective function is of 20 variables.

From the previous analysis, we can summarise as follows:

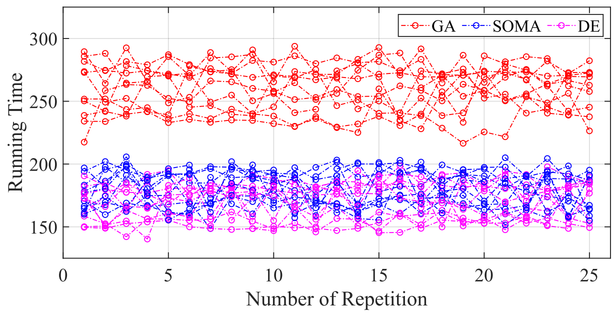
- The application of optimization algorithms has significantly improved the F-transform-based image compression.
- Among the selected optimization methods (SOMA, DE, GA), the GA shows its outperformance in complicated cases where the objective function is of more variables (or the compressed image size is larger).

The second conclusion supports the idea that GA should be used for compressing large images.

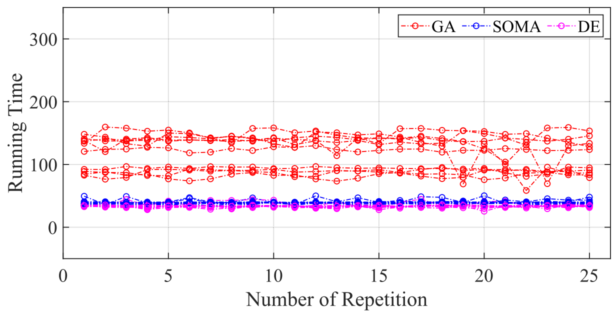
[4]WRT is a nonparametric statistical test that is well-known in the field of numerical optimization for comparing the position of two populations when the observations are in pairs. More details of this test can be found in [44], [45], and [46].



(a) on the 12 × 12 image size



(b) on the 8 × 8 image size



(c) on the 4 × 4 image size

FIGURE 8. Execution time of proposed optimization algorithms performed on 10 selected images.

B. VISUAL COMPARISION

As experimental results in Subsection V-A, the F-transform optimized by the GA algorithm gives the best performance among the selected methods in the cases where the size of compressed images is large. In this subsection, we apply this method to compress and decompress an image of size 128×128 to illustrate the superiority of it that can be visible by eyes in comparison with the original F-transform method.

We select the well-known image "cameraman", convert it to a grayscale image of size 128×128 and consider it as an original image for the illustration. To compare the two mentioned F-transform-based image compression methods, one is optimized by GA (FT-GA) and the other is the original

TABLE 2. The comparison based on RMSE in case the compressed image of size 4 × 4.

No. of Pics	RMSE – 25 runs						
	FT	FT-SOMA		FT-DE		FT-GA	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	0.1883	0.1677	8.4984e-17	0.1677	8.4984e-17	0.1677	8.4984e-17
2	0.2188	0.1695	8.4983e-17	0.1695	8.4983e-17	0.1695	8.4983e-17
3	0.2431	0.1966	5.6655e-17	0.1966	5.6655e-17	0.1966	5.6655e-17
4	0.2686	0.1389	2.8327e-17	0.1389	2.8327e-17	0.1389	2.8327e-17
5	0.1645	0.1632	2.8327e-17	0.1632	2.8327e-17	0.1632	2.8327e-17
6	0.2515	0.2169	2.8327e-17	0.2169	2.8327e-17	0.2169	2.8327e-17
7	0.2111	0.1944	2.8327e-17	0.1944	2.8327e-17	0.1944	2.8327e-17
8	0.1745	0.1343	2.8327e-17	0.1343	2.8327e-17	0.1343	2.8327e-17
9	0.2821	0.1144	4.2491e-17	0.1144	4.2491e-17	0.1144	4.2491e-17
10	0.2217	0.1486	2.8327e-17	0.1486	2.8327e-17	0.1486	5.6655e-17

TABLE 3. The comparison based on RMSE in case the compressed image of size 8 × 8.

No. of Pics	RMSE – 25 runs						
	FT	FT-SOMA		FT-DE		FT-GA	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	0.1420	0.0996 (2)	1.3012e-03	0.0982 (1)	1.4164e-17	0.1002 (2)	1.2011e-03
2	0.1368	0.0993 (1)	2.8025e-04	0.0992 (1)	2.8328e-17	0.1000 (2)	7.0990e-04
3	0.1403	0.1159 (1)	3.4855e-04	0.1159 (1)	3.4855e-04	0.1167 (2)	5.4214e-04
4	0.1045	0.0857 (2)	1.5936e-04	0.0856 (1)	2.8328e-17	0.0858 (3)	1.4164e-17
5	0.1053	0.0887 (2)	9.2876e-04	0.0883 (1)	2.8328e-17	0.0895 (3)	6.6960e-04
6	0.1325	0.1198 (1)	4.6977e-04	0.1196 (1)	4.2492e-17	0.1234 (3)	4.0012e-03
7	0.1296	0.1111 (1)	9.1354e-04	0.1107 (1)	2.8328e-17	0.1112 (1)	1.0001e-03
8	0.0931	0.0770 (3)	6.9568e-04	0.0767 (2)	6.8723e-04	0.0758 (1)	1.4164e-17
9	0.0956	0.0673 (2)	3.7444e-04	0.0670 (1)	2.3165e-04	0.0669 (1)	1.4164e-17
10	0.1216	0.0925 (2)	2.4001e-03	0.0905 (1)	1.4164e-17	0.0909 (1)	8.8453e-04

TABLE 4. The comparison based on RMSE in case the compressed image of size 12 × 12.

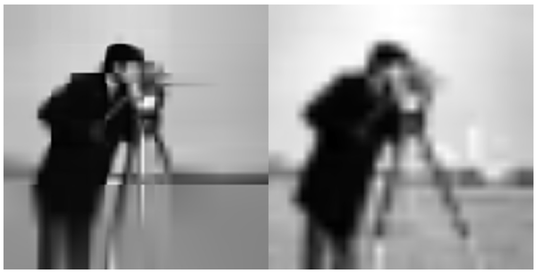
No. of Pics	RMSE – 25 runs						
	FT	FT-SOMA		FT-DE		FT-GA	
		Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	0.0901	0.0561 (2)	0.0038	0.0562 (2)	0.0027	0.0523 (1)	8.4791e-05
2	0.0713	0.0544 (2)	0.0027	0.0589 (3)	0.0044	0.0519 (1)	1.4164e-17
3	0.0932	0.0583 (2)	0.0088	0.0680 (3)	0.0069	0.0507 (1)	1.4164e-17
4	0.0637	0.0479 (2)	0.0080	0.0486 (2)	0.0035	0.0425 (1)	7.0820e-17
5	0.0620	0.0509 (2)	0.0039	0.0519 (2)	0.0037	0.0470 (1)	8.6420e-17
6	0.0936	0.0737 (2)	0.0104	0.0717 (2)	0.0045	0.0635 (1)	3.6039e-04
7	0.0912	0.0643 (2)	0.0051	0.0642 (2)	0.0035	0.0591 (1)	2.6704e-04
8	0.0802	0.0496 (2)	0.0025	0.0525 (3)	0.0021	0.0470 (1)	6.4650e-17
9	0.0642	0.0355 (2)	0.0047	0.0396 (3)	0.0032	0.0311 (1)	7.0820e-17
10	0.0857	0.0579 (2)	0.0054	0.0591 (2)	0.0032	0.0522 (1)	2.8328e-17

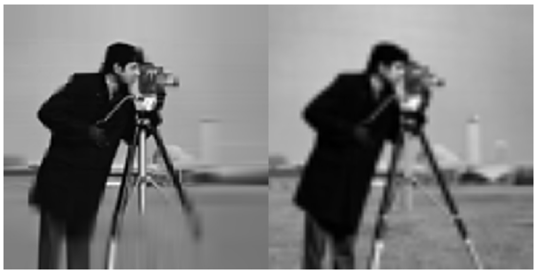
version (FT), we apply them to compress the selected image to images of sizes 32×32, 64×64 and 96×96 (corresponding to the ratio of 1/16, 1/4, and 9/16) and then reconstruct the original image from the compressed ones to evaluate the reconstruction quality (measured by RMSE). As in the experiment in Subsection V-A, the illustrations with GA are repeated 25 times. A comparison between reconstructed images obtained by the two mentioned methods is depicted in Figure 9. In this figure, the results of the GA method are randomly chosen out of 25 runs of the algorithm. There is no doubt that one can recognize the difference between using or not using the GA in the image compression by the F-transform. The average RMSE of 25 running times of the GA method is displayed in Figure 10 together with that obtained by the original one. The compression ratio in this figure is defined as the size ratio of the compressed images to the original.

From the obtained results, we once again see that the application of the optimization algorithms to image compression brings outstanding results not only in terms of data as in Subsection V-A but also in visual evaluation.

VI. CONCLUSION

This paper is a fairly complete and thorough study of the application of popular bio-inspired optimization algorithms, namely the SOMA, DE, and GA, to image compression using the F-transform. To optimize the F-transform-based image compression, we introduced the variable parameterized fuzzy partition with respect to parameters that directly affect the approximation ability of the F-transform. An objective


(a) from the compressed image of size 32×32 (compression ratio is $1/16$)


(b) from the compressed image of size 64×64 (compression ratio is $1/4$)

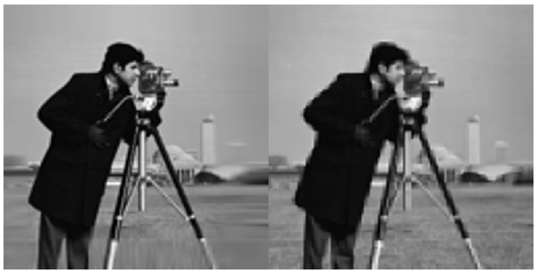

(c) from the compressed image of size 96×96 (compression ratio is $9/16$)

FIGURE 9. The reconstructions by the FT-GA (left) and FT (right) methods.

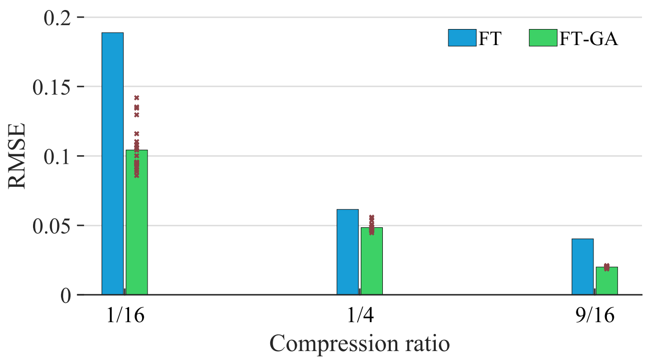


FIGURE 10. The reconstruction quality by the FT-GA and FT methods evaluated by RMSE. The small squares are the RMSE of 25 running times and the green columns are the mean values of them.

function of the image compression was determined according to those parameters for searching optimal solutions. The proposed studies were tested on various images. It shows that the application of optimization algorithms has significantly enhanced the quality of the original F-transform-based image compression. However, the performance of selected algorithms is not the same in this issue. Particularly, the SOMA and DE perform quite well in the case of small compressed image sizes while GA shows the best performance for the bigger sizes.

We restrict the consideration to bio-inspired optimization algorithms. However, today there are quite a few classes of powerful optimization algorithms (especially, physis-based algorithms such as Gravitational Local Search Algorithm (GLSA) [47], Gravitational Search Algorithm (GSA) [16], Big-Bang Big-Crunch (BBBC) [48], and so on [49]) that have proven effective in many areas of application. Therefore, besides the presented studies, it will be interesting if we can extend our consideration to many other classes of optimization algorithms to have a better overview and comparison in optimizing the F-transform-based image compression. This, therefore, is the next interesting topic for our future work.

On the other hand, the positive results in the current paper have opened up a research direction that is to apply optimization algorithms to optimize applications that the F-transform has done very well such as time series decomposition, dimensionality reduction of high-dimensional data, pattern recognition, and many others.

APPENDIX A
LIST OF ABBREVIATIONS

SOMA	Self-Organizing Migrating Algorithm
DE	Differential Evolution
GA	Genetic Algorithm
FT	Fuzzy Transform
FT-SOMA	F-transform-based Image Compression Optimized by the Self-Organizing Migrating Algorithm
FT-DE	F-transform-based Image Compression Optimized by the Differential Evolution
FT-GA	F-transform-based Image Compression Optimized by the Genetic Algorithm
EAs	Evolutionary Algorithms
SI	Swarm Intelligence
CEC	Congress on Evolutionary Computation
FEs	Function Evaluations
RMSE	Root Mean Square Error
WRT	Wilcoxon signed-rank test
GLSA	Gravitational Local Search Algorithm
GSA	Gravitational Search Algorithm
BBBC	Big-Bang Big-Crunch

APPENDIX B
DATA AND SOURCES

All data including optimization algorithms are publicly available at https://github.com/diepquocbao/SI-EA-in-Image-Compression-by-F-transform.

REFERENCES

[1] I. Perfilieva and P. Hurtik, "The F-transform preprocessing for JPEG strong compression of high-resolution images," *Inf. Sci.*, vol. 550, pp. 221–238, Mar. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520310355

[2] H. Lu, Y. Gui, X. Jiang, F. Wu, and C. W. Chen, "Compressed robust transmission for remote sensing services in space information networks," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 46–54, Apr. 2019.

[3] F. Wu, X. Peng, and J. Xu, "LineCast: Line-based distributed coding and transmission for broadcasting satellite images," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 1015–1027, Mar. 2014.

[4] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. London, U.K.: The Institution of Engineering and Technology, 2003.

[5] I. Perfilieva and E. Haldeeva, "Fuzzy transformation," in *Proc. Joint 9th IFSA World Congr. 20th NAFIPS Int. Conf.*, vol. 4, 2001, pp. 1946–1948. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/944365

[6] I. Perfilieva, "Fuzzy transforms: Theory and applications," *Fuzzy Sets Syst.*, vol. 157, no. 8, pp. 993–1023, Apr. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165011405005804

[7] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6313399

[8] I. Perfilieva, "Fuzzy transforms and their applications to image compression," in *Fuzzy Logic and Applications*, I. Bloch, A. Petrosino, and A. G. B. Tettamanzi, Eds. Berlin, Germany: Springer, 2006, pp. 19–31. [Online]. Available: https://link.springer.com/chapter/10.1007/11676935_3

[9] F. Di Martino, V. Loia, I. Perfilieva, and S. Sessa, "An image coding/decoding method based on direct and inverse fuzzy transforms," *Int. J. Approx. Reasoning*, vol. 48, no. 1, pp. 110–131, Apr. 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888613X0700093X

[10] F. Di Martino, P. Hurtik, I. Perfilieva, and S. Sessa, "A color image reduction based on fuzzy transforms," *Inf. Sci.*, vol. 266, no. 2, pp. 101–111, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514000334

[11] P. Hurtik and I. Perfilieva, "Image compression methodology based on fuzzy transform using block similarity," in *Proc. 8th Conf. Eur. Soc. Fuzzy Log. Technol.*, 2013, pp. 561–566, doi: 10.2991/eusflat.2013.79.

[12] P. Hurtik and I. Perfilieva, "A hybrid image compression algorithm based on JPEG and fuzzy transform," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8015614

[13] P. Hurtik and I. Perfilieva, "Noise inuence in FzT+JPEG image compression," in *Proc. Data Sci. Knowledge Eng. Sens. Decis. Support, 13th Int. FLINS Conf.*, 2018, pp. 1433–1439. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789813273238_0178

[14] F. D. Martino, S. Sessa, and I. Perfilieva, "First order fuzzy transform for images compression," *J. Signal Inf. Process.*, vol. 8, no. 3, pp. 178–194, 2017, doi: 10.4236/jsip.2017.83012.

[15] D. Paternain, A. Jurio, J. Ruiz-Aranguren, M. Minarova, Z. Takac, and H. Bustince, "Optimized fuzzy transform for image compression," in *Advances in Fuzzy Logic and Technology*, J. Kacprzyk, E. Szmidt, S. Zadrozny, K. T. Atanassov, and M. Krawczak, Eds. Cham, Switzerland: Springer, 2018, pp. 118–128. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-66827-7_11

[16] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025509001200

[17] A. Kumar, S. Das, and I. Zelinka, "A self-adaptive spherical search algorithm for real-world constrained optimization problems," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 13–14, doi: 10.1145/3377929.3398186.

[18] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9185577

[19] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion Wesley*, vol. 1989, no. 102, p. 36, 1989.

[20] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997. [Online]. Available: https://link.springer.com/article/10.1023/a:1008202821328

[21] Z. Ivan and L. Jouni, "SOMA—Self-organizing migrating algorithm mendel," in *Proc. 6th Int. Conf. Soft Comput.*, Brno, Czech Republic, 2000, pp 167–217.

[22] I. Zelinka, *SOMA—Self-Organizing Migrating Algorithm*. Berlin, Germany: Springer, 2004, pp. 167–217, doi: 10.1007/978-3-540-39930-8_7.

[23] I. Zelinka, *SOMA—Self-Organizing Migrating Algorithm*. Cham, Switzerland: Springer, 2016, pp. 3–49, doi: 10.1007/978-3-319-28161-2_1.

[24] T. Kadavy, M. Pluhacek, A. Viktorin, and R. Senkerik, "Self-organizing migrating algorithm with clustering-aided migration," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 1441–1447, doi: 10.1145/3377929.3398129.

[25] L. Tomaszek, I. Zelinka, and M. Chadli, "On the leader selection in the self-organizing migrating algorithm," *MENDEL*, vol. 25, no. 1, pp. 171–178, Jun. 2019. [Online]. Available: https://mendel-journal.org/index.php/mendel/article/view/95

[26] L. Skanderova, T. Fabian, and I. Zelinka, "Self-adapting self-organizing migrating algorithm," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100593. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650219300756

[27] T. Kadavy, M. Pluhacek, R. Senkerik, and A. Viktorin, "The ensemble of strategies and perturbation parameter in self-organizing migrating algorithm solving CEC 2019 100-digit challenge," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 372–375. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8790012

[28] A. Duary, M. S. Rahman, A. A. Shaikh, S. T. A. Niaki, and A. K. Bhunia, "A new hybrid algorithm to solve bound-constrained nonlinear optimization problems," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12427–12452, Aug. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s00521-019-04696-7

[29] K. Deep and S. Dipti, "A self-organizing migrating genetic algorithm for constrained optimization," *Appl. Math. Comput.*, vol. 198, no. 1, pp. 237–250, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300307008843

[30] L. Skanderova, "Self-organizing migrating algorithm: Review, improvements and comparison," *Artif. Intell. Rev.*, vol. 56, no. 1, pp. 101–172, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10462-022-10167-8

[31] K. Price, N. Awad, M. Ali, and P. Suganthan, "The 2019 100-digit challenge on real-parameter, single objective optimization: Analysis of results," School EEE, Nanyang Technol. Univ., Singapore, Tech. Rep., Dec. 2019. [Online]. Available: https://github.com/P-N-Suganthan/CEC2019

[32] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11042-020-10139-6

[33] M. S. Umam, M. Mustafid, and S. Suryono, "A hybrid genetic algorithm and Tabu search for minimizing makespan in flow shop scheduling problem," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7459–7467, Oct. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821002287

[34] G. Singh, M. Prateek, S. Kumar, M. Verma, D. Singh, and H. Lee, "Hybrid genetic firefly algorithm-based routing protocol for VANETs," *IEEE Access*, vol. 10, pp. 9142–9151, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9681052

[35] J. Luo, S. Fujimura, D. El Baz, and B. Plazolles, "GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem," *J. Parallel Distrib. Comput.*, vol. 133, pp. 244–257, Nov. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731518305628

[36] M. Abbasi, M. Rafiee, M. R. Khosravi, A. Jolfaei, V. G. Menon, and J. M. Koushyar, "An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–14, Dec. 2020. [Online]. Available: https://link.springer.com/article/10.1186/s13677-020-0157-4

[37] G. Fuertes, M. Vargas, M. Alfaro, R. Soto-Garrido, J. Sabattin, and M. A. Peralta, "Chaotic genetic algorithm and the effects of entropy in performance optimization," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 29, no. 1, Jan. 2019, Art. no. 013132. [Online]. Available: https://aip.scitation.org/doi/abs/10.1063/1.5048299

[38] M. Tahir, A. Tubaishat, F. Al-Obeidat, B. Shah, Z. Halim, and M. Waqas, "A novel binary chaotic genetic algorithm for feature selection and its utility in affective computing and healthcare," *Neural Comput. Appl.*, vol. 34, pp. 11453–11474, Sep. 2020. [Online]. Available: https://link.springer.com/article/10.1007/s00521-020-05347-y

[39] S. S. Shreem, H. Turabieh, S. Al Azwari, and F. Baothman, “Enhanced binary genetic algorithm as a feature selection to predict student performance,” *Soft Comput.*, vol. 26, no. 4, pp. 1811–1823, Feb. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s00500-021-06424-7

[40] H. Song, J. Wang, L. Song, H. Zhang, J. Bei, J. Ni, and B. Ye, “Improvement and application of hybrid real-coded genetic algorithm,” *Appl. Intell.*, vol. 52, pp. 17410–17448, Apr. 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10489-021-03048-0

[41] J. Brest, M. S. Maucec, and B. Boskovic, “The 100-digit challenge: Algorithm jDE100,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 19–26. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8789904

[42] V. Stanovov, S. Akhmedova, and E. Semenkin, “NL-SHADE-RSP algorithm with adaptive archive and selective pressure for CEC 2021 numerical optimization,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 809–816. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9504959

[43] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, “Differential evolution: A recent review based on state-of-the-art works,” *Alexandria Eng. J.*, vol. 61, no. 5, pp. 3831–3872, May 2022. [Online]. Available:[Online]. Available: https://www.sciencedirect.com/science/article/pii/S111001682100613X

[44] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*. Boca Raton, FL, USA: CRC Press, 2014.

[45] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650211000034

[46] J. Carrasco, S. García, M. M. Rueda, S. Das, and F. Herrera, “Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review,” *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100665. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650219302639

[47] B. Webster and P. J. Bernhard, “A local search optimization algorithm based on natural principles of gravitation,” Florida Inst. Technol., Melbourne, FL, USA, Tech. Rep. CS-2003-10, 2003. [Online]. Available: http://hdl.handle.net/11141/117

[48] O. K. Erol and I. Eksin, “A new optimization method: Big bang-big crunch,” *Adv. Eng. Softw.*, vol. 37, no. 2, pp. 106–111, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997805000827

[49] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997813001853

NGUYEN LE TOAN NHAT LINH received the Ph.D. degree in applied mathematics and fuzzy modeling from the University of Ostrava, Czech Republic, in 2018. He was a Junior Researcher with the Institute for Research and Applications of Fuzzy Modeling (IRAFM), University of Ostrava, from 2018 to 2021. Since 2021, he has been working as a Lecturer in applied mathematics with the Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam. He has published over 20 papers in conference proceedings and scientific journals. His research interests include fuzzy modeling, data (time series) analysis, applications of soft computing methods in financial modeling, and numerical methods to differential equations.

QUOC BAO DIEP received the B.Eng. degree in mechatronics from Can Tho University, the M.Eng. degree in mechatronics engineering from the Vietnam National University—Ho Chi Minh City University of Technology, and the Ph.D. degree in computer science, communication technology, and applied mathematics from VŠB—Technical University of Ostrava, Czech Republic. He is currently working with the Mechatronics Department, Van Lang University, Vietnam. He has published more than 20 research articles in peer-reviewed journals and international conferences. His research interests include robotics, machine learning, and swarm and evolutionary computation.