Optimized Fuzzy Transform for Image Compression

Daniel Paternain[1,2(✉)], Aranzazu Jurio[1,2], Javier Ruiz-Aranguren[1], Maria Minárová[3], Zdenko Takáč[3], and Humberto Bustince[1,2]

[1] Departamento de Automatica y Computacion, Universidad Publica de Navarra, Campus Arrosadia sn, 31006 Pamplona, Spain
daniel.paternain@unavarra.es
[2] Institute of Smart Cities, Universidad Publica de Navarra, Campus Arrosadia Sn, 31006 Pamplona, Spain
[3] Slovak Univeristy of Technology in Bratislava, Radlinskeho 9, 813 68 Bratislava, Slovakia

Abstract. In this work we propose an image compression algorithm based on the fuzzy transform. The algorithm tries to find the best fuzzy partition of the functions domain in order to obtain the best compressed image (in terms of quality). To solve the optimization problem we based ourselves in the Gravitational Search Algorithm, in which each agent represents a possible fuzzy partition of a fixed size.

Keywords: Fuzzy transform · Image compression · Gravitational Search Algorithm

1 Introduction

Image compression consists in reducing the amount of data (generally measured as the number of bits) required to represent an image [4]. Generally, image compression algorithms transform or encode the image and this transformation is stored or transmitted. Then, an inverse transformation or decode process is applied and a reconstruction of the original image is obtained.

There exist many image compression algorithms in the literature. In this work, we focus on the fuzzy transform [7] which has been successfully applied in the field of image processing and, more specifically, in image compression (see [1–3,6,8,9]). Broadly speaking, the fuzzy transform is based on a fuzzy partition of the image domain, i.e. $[1, N] \times [1, M]$ where N and M represent, respectively, the number of rows and columns of an image. The fuzzy partition is composed by n and m fuzzy sets defined on the intervals $[1, N]$ and $[1, M]$, respectively, with certain properties. It is known that, given fixed n and m, two different fuzzy partitions of the image domain will yield two different fuzzy transforms, two different compressed images and, accordingly, two different reconstructed images. Since the inverse fuzzy transform is also based on the same fuzzy partition used

J. Kacprzyk et al. (eds.), *Advances in Fuzzy Logic and Technology 2017*,
Advances in Intelligent Systems and Computing 643,
DOI 10.1007/978-3-319-66827-7_11

in the compression, it is possible to rank both fuzzy partitions in terms of quality: the better the quality of the reconstructed image, the better the fuzzy partition.

Taking into account this fact, the objective of this work is the following: given an image, to find the best fuzzy partition so that the reconstructed image after applying the fuzzy and the inverse fuzzy transform is as similar as possible to the original image.

In order to find the best fuzzy partition, we propose to use the Gravitational Search Algorithm (GSA), an heuristic optimization algorithm based on the law of gravity and the law of motion [10]. In this algorithm, each agent represents a solution, i.e. a feasible fuzzy partition of the image domain. The quality (fitness) of an specific agent is measured in the following way: taking the fuzzy partition represented by the agent, we apply the fuzzy transform (obtaining the compressed image) and, later, the inverse fuzzy transform. Finally, we measure the quality of the reconstructed image by means of an error measure and we associate this error with the agent. Then, the problem becomes a minimization problem and the GSA tries to find the agent with the minimum error measure.

The structure of this work is as follows. In Sect. 2 we recall the concept of a fuzzy partition and the definition of the fuzzy and inverse fuzzy transform. In Sect. 3 we summarize the steps of the Gravitational Search Algorithm. In Sect. 4 we explain in detail the optimization algorithm we propose in order to find the best fuzzy partition and, in Sect. 5, we show the first preliminary results obtained by our proposal. We finish, in Sect. 6 with some conclusions and future research lines.

2 Fuzzy Transform of Discrete Functions

In this section we recall the concept of the fuzzy transform and the inverse fuzzy transform. For the sake of simplicity, we focus only on the discrete fuzzy transform, that maps a discrete function defined on an interval of real numbers into a real vector.

In order to give the definition of the fuzzy and inverse fuzzy transform, we define the concept of a fuzzy partition of the functions domain.

Definition 1 [7]. *Let $x_1 < \cdots < x_n$ be fixed nodes within $[a, b]$, such that $x_1 = a$, $x_n = b$ and $n \geq 2$. We say that fuzzy sets $A_1, \ldots, A_n$, identified with their membership functions $A_1(x), \ldots, A_n(x)$ defined on $[a, b]$, form a fuzzy partition of $[a, b]$ if they fulfill the following conditions for $k = 1, \ldots, n$:*

(1) $A_k : [a, b] \to [0, 1]$, $A_k(x_k) = 1$;
(2) $A_k(x) = 0$ if $x \notin (x_{k-1}, x_{k+1})$ where $x_0 = a$ and $x_{n+1} = b$;
(3) $A_k(x)$ is continuous;
(4) $A_k(x)$, $k = 2, \ldots, n$, strictly increases on $[x_{k-1}, x_k]$ and $A_k(x)$, $k = 1, \ldots, n-1$, strictly decreases on $[x_k, x_{k+1}]$;
(5) for all $x \in [a, b]$, $\sum_{k=1}^{n} A_k(x) = 1$.

Definition 2 [7]. *Let a fuzzy partition of $[a,b]$ be given by fuzzy sets $A_1,\dots,A_n$ in the sense of Definition 1. We say that it is uniform if the nodes $x_1,\dots,x_n$, $n \geq 3$, are equidistant. This means that $x_k = a + h(k-1)$, $k = 1,\dots,n$, where $h = (b-a)/(n-1)$, and*

(6) $A_k(x_k - x) = A_k(x_k + x)$, for all $x \in [0,h]$, $k = 2,\dots,n-1$;
(7) $A_k(x) = A_{k-1}(x-h)$, for all $k = 2,\dots,n-1$ and $x \in [x_k, x_{k+1}]$, and $A_{k+1}(x) = A_k(x-h)$, for all $k = 2,\dots,n-1$ and $x \in [x_k, x_{k+1}]$.

Definition 3 [7]. *Let f be a function given at nodes $p_1,\dots,p_l \in [a,b]$ and $A_1,\dots,A_n$, $n < l$, be basic functions which form a fuzzy partition of $[a,b]$. We say that the n-tuple of real numbers $F[f] = (F_1,\dots,F_n)$ given by*

$$F_k = \frac{\sum_{j=1}^{l} f(p_j) A_k(p_j)}{\sum_{j=1}^{l} A_k(p_j)}, \quad k = 1,\dots,n \tag{1}$$

is the (discrete) fuzzy transform of f with respect to $A_1,\dots,A_n$.

Definition 4 [7]. *Let f be given at nodes $p_1,\dots,p_l \in [a,b]$ and $F[f]$ be the fuzzy transform of f with respect to the fuzzy partition $A_1,\dots,A_n$. Then, the function*

$$f_F(p_j) = \sum_{k=1}^{n} F_k A_k(p_j)$$

defined at the same nodes is the inverse fuzzy transform.

The definition of the fuzzy and inverse fuzzy transform can be extended to functions of more than one variable. In this work we are interested in working with images, which can be seen as discrete functions of two variables. Therefore, we will use the two-dimensional fuzzy and inverse fuzzy transform.

Definition 5 [7]. *Let a function f be given at nodes $(p_i,q_j) \in [a,b] \times [c,d]$, $i = 1,\dots,N$, $j = 1,\dots,M$ and $A_1,\dots,A_n$, $B_1,\dots,B_m$ where $n < N$, $m < M$, be basic functions which form a fuzzy partition of $[a,b]$ and $[c,d]$, respectively. Suppose that sets $p_1,\dots,p_N$ and $q_1,\dots,q_M$ of these nodes are sufficiently dense with respect to the chosen partition. We say that the $n \times m$ matrix of real numbers $\mathbf{F}[f] = (F_{kl})$ is the discrete fuzzy transform of f with respect to $A_1,\dots,A_n$, and $B_1,\dots,B_m$ if*

$$F_{kl} = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} f(p_i,q_j) A_k(p_i) B_l(q_j)}{\sum_{j=1}^{M}\sum_{i=1}^{N} A_k(p_i) B_l(q_j)} \tag{2}$$

holds for all $k = 1,\dots,n$, $l = 1,\dots,m$.

Definition 6 [7]. *Let $A_1,\dots,A_n$ and $B_1,\dots,B_m$ be basic functions which form a fuzzy partition of $[a,b]$ and $[c,d]$ respectively. Let f be given at points $(p_i,q_j) \in [a,b] \times [c,d]$, $i = 1,\dots,N$, $j = 1,\dots,M$ and $\mathbf{F}[f]$ be the fuzzy transform of f with respect to $A_1,\dots,A_n$ and $B_1,\dots,B_m$. Then the function*

$$f_{\mathbf{F}}(p_i,q_j) = \sum_{k=1}^{n}\sum_{l=1}^{m} F_{kl} A_k(p_i) B_l(q_j) \tag{3}$$

defined at the same nodes is the inverse fuzzy transform.

3 The Gravitational Search Algorithm

In this section we present the main steps concerning the GSA. Basically, the objective of the GSA is to minimize (or maximize) a fitness function defined on an n-dimensional space (search space). The optimization is performed iteratively by means of a set of agents that represent feasible solutions. Each agent has an acceleration and a velocity that is determined by the effect of the rest of agents: the best agents (the best solutions to the optimization problem) attract each other with a greater force.

Technically, consider a systems of T particles (agents). The position of each agent X_i in a p-dimensional space is given by

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^p)$$

for each $i \in \{1, \ldots, T\}$. At each specific time t, the mass of a particle X_i represents the adaptation of that specific particle to the problem. This is done by means of a fitness function, that maps the p-dimensional space where the particles are defined into the set of real positive numbers. Each agent X_i has a particular mass, which is determined by the fitness function as follows:

$$m_i(t) = \frac{fitness_i(t) - worst(t)}{best(t) - worst(t)}$$

where

$$best(t) = \min_{j \in \{1,\ldots,T\}} fit_j(t) \text{ and}$$

$$worst(t) = \max_{j \in \{1,\ldots,T\}} fit_j(t)$$

Finally, the masses are normalized by means of:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{T} m_j(t)}$$

such that $\sum_{i=1}^{T} M_i(t) = 1$.

Remark 1. Previous formulae are used in minimization problems. In the case of a maximization problem, $best(t)$ and $worst(t)$ are calculated as the maximum and minimum of fitness function, respectively.

As commented before, the GSA is based on the movement of agents, that search along the search space for minima of the fitness function. In order to get this movement, an acceleration is calculated for each agent as the result of the acting forces of the rest of agents. The force acting on agent i by the rest of agents is given as

$$F_i^d(t) = \sum_{j=1 j \neq i}^{T} r_j F_{ij}^d(t)$$

where r_j is a random number in $[0, 1]$ and

$$F_{ij}^d(t) = G(t)\frac{M_i(t)M_j(t)}{R_{ij}(t) + \epsilon}(x_j^d(t) - x_i^d(t)),$$

ϵ is small positive constant and

$$R_{ij}(t) = ||X_i(t) - X_j(t)||_2.$$

Once the force acting over agent i is calculated, the acceleration is given by

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j=1 j \neq i}^{T} r_j \frac{M_j(t)}{R_{ij}(t)}(x_j^d(t) - x_i^t(d).$$

The next velocity of an agent is given as

$$v_i^d(t+1) = r_i v_i^d(t) + a_i^d(t)$$

where r_i is a random number in $[0, 1]$ and, finally, the next position is given as

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1).$$

Taking into account previous formulae, the most suitable agents will tend to have heavier masses along the iterations and, therefore, tend to attract the other with greater forces. At the end, the agents tend to move toward the best agent. A summarization of the GSA is shown in Algorithm 1.

Algorithm 1. Gravitational Search Algorithm

Input: Number of agents T. Fitness function.
Output: Best agent
  for $i = 1, \ldots, T$ **do**
    Random initialization of X_i
  end for
  while stop criteria not reached **do**
    for $i = 1, \ldots, T$ **do**
      Evaluate fitness $fitness_i(t)$ of each agent X_i
    end for
    Update $G(t), best(t), worst(t)$ and $M_i(t)$
    for $i = 1, \ldots, T$ **do**
      Calculate the total force $F_i(t)$ acting on agent i
      Calculate the acceleration $A_i(t)$
      Calculate the velocity $V_i(t)$
      Update position $X_i(t+1)$
    end for
  end while
  Return $best(t)$ as the best agent

4 GSA for Tuning the Fuzzy Partition of the Fuzzy Transform

As we have stated in the introduction, the fuzzy (and inverse) transform have demonstrated to be a useful tool for image compression. Usually, the use of a uniform fuzzy partition is assumed when compressing images due to its simplicity and the fact that no extra information needs to be stored. However, it is clear that given a fixed number of nodes (fuzzy sets) in the fuzzy partition, different locations of nodes produce different fuzzy transforms. In this section we explain how to optimize the location of the nodes of a fuzzy partition in order to obtain better compressed images.

We recall that, given an image f of $N \times M$ pixels (generally $f : \{1, \ldots, N\} \times \{1, \ldots, M\} \to \{0, 1, \ldots, 255\}$) and fuzzy sets $A_1, \ldots, A_n : [1, N] \to [0, 1]$, $B_1, \ldots, B_m : [1, M] \to [0, 1]$ forming a fuzzy partition of $[1, N]$ and $[1, M]$, respectively, the fuzzy transform $\mathbf{F}$ of f with respect to $A_1, \ldots, A_n$, $B_1, \ldots, B_m$ is a new (compressed) matrix of $n \times m$ pixels. The inversion process is performed by the inverse fuzzy transform that, starting from $\mathbf{F}$ and from the same fuzzy partition $A_1, \ldots, A_n$, $B_1, \ldots, B_m$, obtains a new (uncompressed) matrix $f_{\mathbf{F}}$ of $N \times M$ pixels.

The loss of quality produced by applying the fuzzy and inverse fuzzy transform to image f can be calculated by means of several distance or error measures. One of such measures is the main squared error (MSE), that is calculated as follows:

$$MSE(f, f_{\mathbf{F}}) = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (f(p_i, q_j) - f_{\mathbf{F}}(p_i, q_j))^2.$$

Since both $\mathbf{F}$ and $f_{\mathbf{F}}$ depends on the fuzzy partition formed by $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$, one possible way of optimizing the fuzzy transform is by tuning the position of the fuzzy subsets A_i, B_j [11].

Remark 2. In this work we assume that every fuzzy set of the fuzzy partition has triangular membership function. Therefore, a fuzzy partition is totally determined by the position of the fixed nodes (see Definition 1).

In order to do this, we propose to model each possible fuzzy partition by means of a real vector of $n+m$ real numbers. The first n real numbers represent the position of the fixed nodes $x_1, \ldots, x_n \in [1, N]$ such that $A_i(x_i) = 1$. The real numbers going from the $n+1$-th to the $n+m$-th position represent the nodes $y_1, \ldots, y_m \in [1, M]$ such that $B_j(y_j) = 1$. Therefore, a fuzzy partition is represented by a real vector $X \in [1, N]^n \times [1, M]^m$. However, in order to have a valid representation of a fuzzy partition we need to assure that:

- $x_1^j = 1, x_n^j = N, x_{n+1}^j = 1, x_{n+m}^j = M$, for every $j \in \{1, \ldots, p\}$;
- $x_i^j < x_i^{j+1}$ for every $j \in \{1, \ldots, n-1\}$ and every $j \in \{n+1, \ldots, n+m-1\}$;
- $x_i^{j+1} - x_i^{j+1}$ for every $j \in \{2, \ldots, n-1\}$ and every $j \in \{n+2, \ldots, n+m-1\}$.

Remark 3. This process is done in our implementation of the GSA and those agents that represent invalid fuzzy partitions automatically transformed in order to assure previous conditions.

Finally, the summarization of our optimization process is explained in Algorithm 2.

Algorithm 2. Optimization of fuzzy transform

Input: Image f of $N \times M$ pixels. Size of compressed image $n < N$ and $m < M$. Number of agents T.

Output: Optimized fuzzy partition $A_1, \ldots, A_n$, $B_1, \ldots, B_m$. Optimized compressed image $\mathbf{F}$. Uncompressed image $f_{\mathbf{F}}$. MSE between f and $f_{\mathbf{F}}$

for $i = 1, \ldots, T$ **do**

Initialize agent $X_i \in [1, N]^n \times [1, M]^m$ randomly following the restrictions mentioned above.

end for

Execute Algorithm 1 and obtain $Xbest$ as the best particle

Decode $Xbest$ into fuzzy partition $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$

Calculate fuzzy transform $\mathbf{F}$ of f with respect to $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$

Calculate inverse fuzzy transform $f_{\mathbf{F}}$ of F with respect to $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$

Calculate $MSE(f, f_{\mathbf{F}})$

5 Experimental Results

In this section we evaluate our optimization procedure based on the GSA for tuning the fuzzy partition associated with the fuzzy transform. We have first taken an original image of size 321×481 pixels (see Fig. 1) and we have fixed $n = 160, m = 240$. In order to test different settings of the GSA, we evaluate Algorithm 2 with different number of agents, specifically $T = 5, 25, 50, 100$. The MSE of each optimized fuzzy transform is shown in Table 1 (second row). According to the results it is not until we have 100 agents that we outperform the non-optimized uniform fuzzy partition. Besides, the improvement is very small, probably due to the large size of the fuzzy partition (big number of nodes).

With the purpose of testing whether a smaller number of nodes in the fuzzy partition increases the improvement of the optimized fuzzy transform, we have executed Algorithm 2 with

- $n = 107, m = 160$;
- $n = 80, m = 120$ and
- $n = 64, m = 96$.

All of these executions have been also tried with $T = 5, 25, 50, 100$. The MSEs obtained are shown again in Table 1, row number three, four and five, respectively.

Analyzing the results we first realize that having a big number of agents guarantees finding a good enough solution to the optimization problem. If we

Fig. 1. Original image 3096 from [5] (321 × 481 pixels).

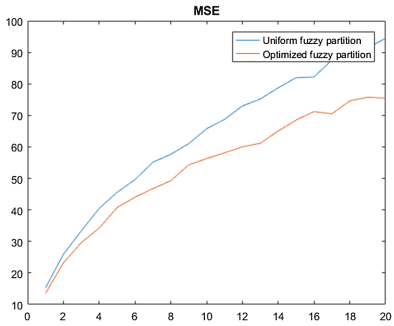

Fig. 2. Comparison of MSE of reconstructed images using a uniform fuzzy partition and Algorithm 2 with $T = 100$.

Table 1. MSE obtained from reconstructed images using uniform and optimized fuzzy partitions.

	Uniform	$T = 5$	$T = 25$	$T = 50$	$T = 100$
$n = 160, m = 240$	12,25	35,40	13,44	13,12	10,62
$n = 107, m = 160$	24,57	68,57	24,09	23,75	21,30
$n = 80, m = 120$	35,56	97,91	37,18	33,09	31,63
$n = 64, m = 96$	46,19	75,74	42,97	41,00	40,86

take $T = 5$ then Algorithm 2 is not able to find a good solution to the problem. However, when $T = 100$ we always outperform the uniform partition.

Now, if we focus on the behavior of the MSE among different partition sizes, we see that the proposed optimization algorithm is able to find better compressed images (if we have enough number of agents). Although the numbers shown in Table 1 do not differ too much depending on the size of the partition (the improvement is around 12% in every case), it seems evident that as long as the size of the compressed image decreases, the improvement should increase. In order to prove this we have executed Algorithm 2 with $T = 100$ and different sizes of the compressed image. The results in terms of MSE are shown in Fig. 2, where the MSE of a uniform and an optimized fuzzy partition are shown. In the horizontal axis we show the different compression rates, i.e. the ratio between

the size of the compressed image (size of the fuzzy partition) and the size of the original image. In this sense, a compression rate of 100 means that the original image has 381×421 and the compressed image 38×42. Observe that the difference between the MSE of the uniform and optimized fuzzy partition increases as long as the compressed ratio increases.

Finally, in Fig. 3 we show the reconstructed images obtained from a uniform (first column) and the reconstructed images obtained from an optimized fuzzy

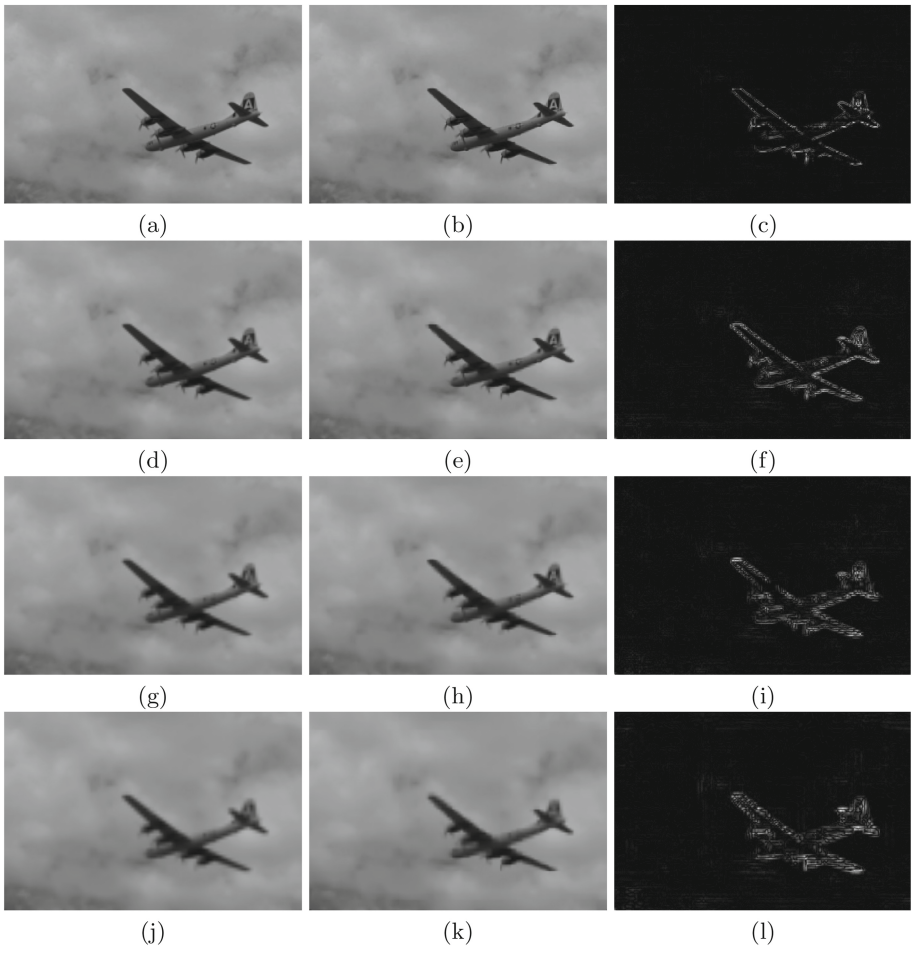

Fig. 3. First column: reconstructed images using a uniform fuzzy partition and sizes $m = 160, n = 240$ (a), $m = 107, n = 160$ (d), $m = 80, n = 120$ (g) and $m = 64, n = 96$ (j). Second column: reconstructed images using an optimized fuzzy partition by Algorithm 2 and sizes $m = 160, n = 240$ (b), $m = 107, n = 160$ (e), $m = 80, n = 120$ (h) and $m = 64, n = 96$ (k). Third column: differences between images a and d (c), d and e (f), g and h (i) and j and k (l).

partition (second column) obtained applying Algorithm 2 and taking $T = 100$ agents and the same sizes used in Table 1. Notice that the differences between the first and second row appears mainly on the edges of objects. This can be better seen in the third column of Fig. 3, where the differences between the images of the first and second column are shown (the differences have been normalized so that the maximum difference appears in white). The conclusion obtained is that while using uniform fuzzy partitions produce blurring in the reconstructed images, the optimization process allows to allocate the fuzzy sets in those areas of interest where there exist large enough changes of intensities. Then, the fuzzy transform is able to capture these changes in a better way and the information is not lost in the process.

6 Conclusions

in this work we have proposed an optimization problem to find the best fuzzy partition associated with an image. The solution of this problem allows to obtain better compressed images by means of the fuzzy transform.

In order to solve the optimization problem we have based on the Gravitational Search Algorithm. The first results obtained show that the GSA is able to obtain optimized fuzzy partitions that minimize the error measure of the image compression procedure.

In the future, we want to extend the experimental study to a wider set of images, analyzing which images are suitable for this algorithm. Moreover, it would be interesting to compare the results when using different optimization algorithms rather than the GSA.

Acknowledgment. This work has been partially supported by MINECO, AEI/FEDER, UE under project TIN2016-77356-P and grant VEGA 1/0420/15.

References

1. Beliakov, G., Bustince, H., Paternain, D.: Image reduction using means on discrete product lattices. IEEE Trans. Image Process. **21**, 1070–1083 (2012)
2. Di Martino, F., Sessa, S.: Compression and decompression of images with discrete fuzzy transforms. Inf. Sci. **177**, 2349–2362 (2007)
3. Di Martino, F., Loia, V., Perfilieva, I., Sessa, S.: An image coding/decoding method based on direct and inverse fuzzy transforms. Int. J. Approx. Reason. **48**, 110–131 (2008)
4. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Pearson International Edition, New Jersey (2008)
5. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the 8th International Conference on Computer Vision, pp. 416–423 (2001)
6. Paternain, D., Fernandez, J., Bustince, H., Mesiar, R., Beliakov, G.: Construction of image reduction operators using averaging aggregation functions. Fuzzy Sets Syst. **261**, 87–111 (2015)

7. Perfilieva, I.: Fuzzy transforms: theory and applications. Fuzzy Sets Syst. **157**, 993–1023 (2006)
8. Perfilieva, I., De Baets, B.: Fuzzy transforms of monotone functions with application to image compression. Inf. Sci. **180**, 3304–3315 (2010)
9. Perfilieva, I., Hurtik, P., Di Martino, F., Sessa, S.: Image reduction method based on the F-transform. Soft Comput. **21**, 1847–1861 (2017)
10. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
11. Štěpnička, S., Polakovič, O.: A neural network approach to the fuzzy transform. Fuzzy Sets Syst. **160**, 1037–1047 (2009)