Revista Brasileira de Computação Aplicada, April, 2025

DOI: 10.5335/rbca.v17i1.16442
Vol. 17, Nº 1, pp. 33–44
Homepage: seer.upf.br/index.php/rbca/index

ORIGINAL PAPER

Fireworks Algorithm and Particle Swarm Optimization for Watermarking in Image Vector Quantization

Marcos José Canêjo[1], Jair Galvão[2], Waslon T. A. Lopes[3], Hugerles S. Silva[4], Francisco Madeiro[5]

[1]Catholic University of Pernambuco - Brazil, [2]Federal Institute of Pernambuco - Brazil, [3]Federal University of Paraíba - Brazil, [4]University of Brasilia - Brazil, [5]Polytechnic School of Pernambuco, University of Pernambuco - Brazil

*marcos.azevedo@unicap.br; jair.araujo@garanhuns.ifpe.edu.br; waslon@cear.ufpb.br; hugerles.silva@ave.it.pt; madeiro@poli.br

Received: 2024-10-29. **Revised:** 2025-04-25. **Accepted:** 2025-04-30.

Abstract

Information hiding has been addressed in several studies. When the information is hidden in digital images, it can be carried out in three domains: spatial domain (original space of image pixels), frequency domain (or transformed domain, such as Discrete Cosine Transform domain) and compressed domain. In the last case, one can cite data embedding in images compressed by vector quantization (VQ). This paper addresses the problem of codebook partition in the scenario of invisible watermark embedding in digital images compressed by VQ. In this paper, two techniques are investigated for partitioning purposes: the PSO (Particle Swarm Optimization) algorithm and the EFA (Enhanced Fireworks Algorithm) as alternatives to the Genetic Algorithm. The performance of the techniques is evaluated with regard to the algorithms execution time. Robustness of the watermark against a variety of attacks is assessed for codebooks partitioned with the aforementioned algorithms.

Keywords: Fireworks Algorithm; Information Hiding; Particle Swarm Optimization; Vector Quantization; Watermarking.

Resumo

A ocultação de informações tem sido abordada em vários estudos. Quando a informação é ocultada em imagens digitais, pode ser realizada em três domínios: domínio espacial (espaço original dos pixels da imagem), domínio da frequência (ou domínio transformado, como o domínio da Transformada Cosseno Discreta) e domínio da imagem comprimida. No último caso, pode-se citar a inserção de dados em imagens comprimidas por quantização vetorial (QV). Este artigo aborda o problema da partição do dicionário no cenário de inserção de marca d'água invisível em imagens digitais comprimidas por QV. Neste artigo, duas técnicas são investigadas para fins de partição: o algoritmo PSO (Particle Swarm Optimization) e o EFA (Enhanced Fireworks Algorithm) como alternativas ao Algoritmo Genético. O desempenho das técnicas é avaliado com relação ao tempo de execução dos algoritmos. A robustez da marca d'água contra uma variedade de ataques é avaliada para dicionários particionados com os algoritmos mencionados.

Palavras-Chave: Algoritmo Fireworks; Ocultação de Informação; Otimização por Enxame de Partículas; Quantização Vetorial; Marca D'água.

1 Introduction

With the growth of internet use, creation, replication and transmission of digital content have become quite common, mainly due to the cheapening of devices, storage facility and a greater availability of bandwidth Evsutin et al. (2020). Thus, the development of solutions for copyright protection, identification of properties of digital media and secure information transmission has become important Fei et al. (2022); Sanivarapu et al. (2022). Besides, data hiding techniques are intended to embed information in audio, video or images, for example.

The watermark techniques Wang et al. (2022); Evsutin et al. (2020); Su et al. (2020) focus on the object used to embed data, aiming, for instance, the detection of violation of integrity or to assure the copyright of the object. With the vast amount of digital content nowadays, the need for the use of digital signal compression techniques is unquestionable. In this context, the signal compression techniques aim to reduce the number of bits to represent digital signals, for the purpose of efficient transmission and/or storage.

Vector Quantization (VQ) Gersho and Gray (2012) is a lossy signal compression technique which uses blocks of samples instead of individual samples and the performance of VQ depends on the designed codebooks. The most widely known algorithm for codebook design is the Linde-Buzo-Gray (LBG) Linde et al. (1980). Other examples of codebook design algorithms are competitive learning algorithms Bispo Jr et al. (2010), fuzzy algorithms Mata et al. (2016), swarm algorithms Severo et al. (2016); Fonseca et al. (2018); Severo et al. (2024), memetic algorithms Azevedo et al. (2009) and deep learning techniques Jiang et al. (2017).

A watermark can be embedded in images compressed by VQ. The embedding is carried out in the indices of the codevectos. The VQ-based watermark technique used in this paper requires a division of the codebook in two groups (partitioning step): one used to embed the bit 0 and the other used to embed the bit 1. In this paper we evaluate the use of algorithms Particle Swarm Optimization (PSO) Kennedy and Eberhart (1995) and Fireworks Algorithm (FA) Tan and Zhu (2010); Zheng et al. (2013) as alternatives to Genetic Algorithm (GA) Goldberg (1989) applied to the image watermarking method introduced by Wang et al. (2007).

In watermark literature, we can mention different methods that use techniques of artificial intelligence (IA) and particularly Deep Neural Network (DNN) to embed the watermark in the spatial and frequency domains Amrit and Singh (2022).

For example, Issa (2018) used two artificial intelligence techniques, Genetic Algorithm (GA) and Cuckoo Search (CS) Joshi et al. (2017), to optimize Scale Factors (SFs) selection in order to enhance the robustness of the watermark. GA's population consisted of a collection of SFs that were measured under mixed rotation, resizing, and average filtering attacks. In the CS version, the original image was divided into four blocks, as it was considered that optimal SF values may vary by region.

Unlike classic watermarking techniques based on spatial domain, Sy et al. (2020) present a digital watermarking scheme for images using a combination of Discrete Wavelet Transform (DWT) and a Convolutional Neural Network (CNN). The process involves transforming the host image into the DWT domain and then using it to train the CNN. The results indicate that the proposed scheme performs well against JPEG compression, average and median filtering, "salt and pepper" noise, Gaussian noise, speckle noise, brightness modification, scaling, cropping, rotation, and shear operations. The robustness of the scheme against these attacks is a positive aspect highlighted by the authors.

In this paper, we evaluate the performance of PSO and FA in the optimization problem of partitioning the VQ codebook for the purpose of inserting a watermark in an image compressed by vector quantization. Specifically, the aforementioned algorithms are compared with the genetic algorithm used in the watermarking technique under consideration. A comparison analysis is proposed in terms of the execution time of the algorithms and the robustness against a variety of attacks.

The remainder of this paper is organized as follows. Section 2 presents the principles of Vector Quantization. Section 3 describes the watermark technique using VQ. Section 4 details the Genetic Algorithm, Particle Swarm Optimization and Fireworks Algorithm applied to codebook partitioning for image watermarking. Section 5 presents the methodology and results. Conclusions are presented in Section 6.

2 Vector Quantization

Vector quantization is the mapping of a K-dimensional vector x in a vector belonging to a finite subset W Gersho and Gray (2012); Gray (1984); Camacho-Gonzalez et al. (2025), which is called a codebook. In other words, VQ performs a mapping $Q : \mathbb{R}^K \rightarrow W$. The codebook $W = \{w_i; i = 1, 2, \ldots, N\}$ is a finite subset of $\mathbb{R}^K$ in which each vector is called codevector. The number of samples (components) of each vector is the dimension (K). The codebook size is the number of codevectors, denoted by N.

Image VQ consists of mapping each block of pixels of the original image to the most similar codevector, the one that has the smallest distance to the original block. Fig. 1 illustrates an example of VQ applied to digital image. In this example, the dimension K of the block is equal to 16 (blocks of 4×4 pixels) with 8 codevectors in the codebook ($N = 8$). It is possible to observe that VQ is a lossy compression technique, *i.e.*, the quantized image is different from the original one. The performance of vector quantization is related to codebook design.

The quality of the reconstructed image can be measured by metrics such as Structural Similarity Index Wang et al. (2004) and Peak Signal-to-Noise Ratio (PSNR), given by

$$\text{PSNR} = 10 \log_{10} \frac{V_p^2}{\text{MSE}}, \tag{1}$$

in which V_p represents the peak amplitude value of the input image and MSE is the mean squared error between the original and the compressed images.

The Linde-Buzo-Gray (LBG) algorithm Linde et al. (1980) is a very popular technique for codebook design. Let n denote the number of iterations of LBG. Given the dimension K (that is, the number of components of the codevectors), the codebook size N, and a distortion threshold δ, the LBG consists of the steps shown in Algorithm 1.

It is worth mentioning that the initial codebook can be obtained, for instance, from a random choice of vectors from the training set. The LBG algorithm aims to reduce the distortion introduced when representing the training vectors by their most similar codevectors.

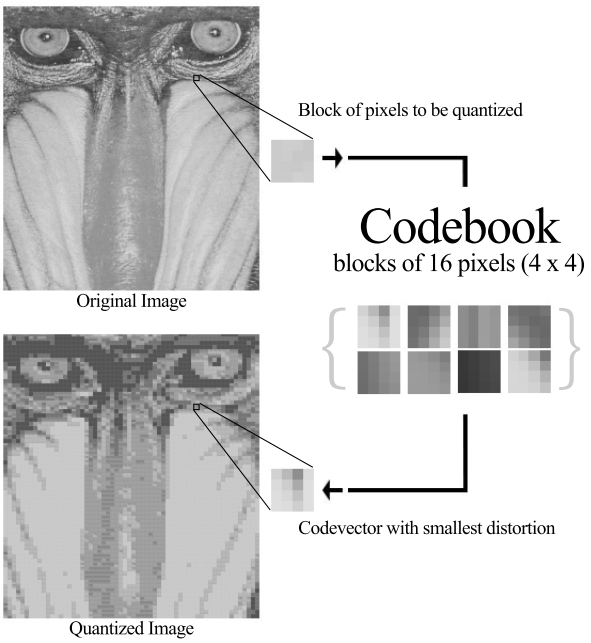

Figure 1: Example of VQ in digital image.

Algorithm 1 LBG Algorithm

1: **Initialization:** begin with an initial codebook $W_0 = \{w_1, w_2, \ldots, w_N\}$, in which N is the number of codevectors, and a training set $X = \{x_m; m = 1, 2, \ldots, M\}$, $j = 0$ and $D_{-1} = \infty$ (a very large number), in which D_j is the distortion in the j-th iteration.

2: **Partitioning:** For W_j (codebook in j-th iteration), allocate each training vector into a region according to the nearest neighbour rule;

3: **Codebook update:** Calculate the new codevectors as the centroids of regions, $w_i = \frac{1}{R_i} \sum_{x_m \in V_i} x_m, 1 \leq i \leq N$, where R_i represents the number of training vectors allocated in the region $V_i = \{x \mid d(x, w_i) < d(x, w_a) \forall a \neq i\}$;

4: **Convergence test:** if $\frac{D_{j-1} - D_j}{D_j} \leq \delta$ stop, W_i representing the final codebook (codebook designed); Otherwise go to Step 2 and increase the iteration counter j.

3 Digital watermark based on vector quantization

Watermarking is a technique that hides information (such as a logo, a mark or a code identifier on a sequence of bits) in medias with the objective, for example, of copyright or to check the integrity of the digital content. Watermarking can be classified according to its robustness and visibility, as depicted in Fig. 2. There are three main requirements under the watermark: fidelity, robustness and payload Evsutin et al. (2020).

The fidelity or transparency is the similarity between the original object and the marked object. The watermark

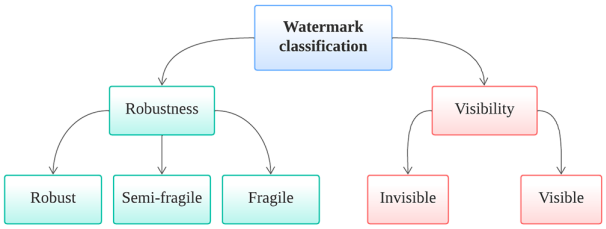

Figure 2: Watermark classification.

should affect as little as possible the quality of the marked image. This applies to the invisible watermark. Robustness refers to the ability of a digital watermark to remain detectable despite various attacks to the watermarked image. For digital images, attacks involve, for instance, cropping, rotation and compression of the watermarked image Roy et al. (2018). For purposes of proof of authorship (copyrights), the watermark should be robust against manipulations. Payload means the number of bits that can be hidden in the image. The larger the size of the object, the greater is its capacity to store bits in the digital content.

3.1 Watermark based on codebook partition

The method of watermark based on VQ considered in this work was introduced in Wang et al. (2007). Precisely, the method consists in dividing the codebook W in two sub-codebooks, G_0 and G_1, using a key C, where $C = \{c_1, c_2, \ldots, c_N\} \mid c_i \in \{0, 1\}, 1 \leq i \leq N$. The sub-codebooks G_0 and G_1 hide the bits 0 and 1, respectively. The Algorithm 2 shows the sequence of steps in the process of inserting the watermark based on VQ.

Algorithm 2 Inserting the watermark

1. Split the codebook W in two sub-codebooks, G_0 and G_1, using the key C;
2. Divide the original image $X = \{x_1, x_2, \ldots, x_T\}$ in blocks of size K pixels, where T is the number of vectors (i.e., number of blocks of K pixels);
3. To the input vector x_i and the bit y of the watermark to be inserted, $y \in \{0, 1\}$, obtain the codevector in the sub-codebook, G_0 or G_1, with the shortest distance to the input vector. Precisely, if $y = 0$, then the search for the codevector with the shortest distance is carried out in G_0; if $y = 1$, that search is carried out in G_1.
4. Obtain the marked vector x'_i, which is the codevector with the shortest distance to x_i;
5. Repeat steps 3-4 until all bits of the watermark are inserted;
6. Obtain the watermarked image X' composed by all output vectors $\{x'_1, x'_2, \ldots, x'_T\}$.

The steps to perform the extraction of hidden bits of watermark are described in Algorithm 3.

The search for the nearest neighbor has high

complexity. To minimize this complexity, Partial Distortion Search (PDS) Bei and Gray (1985) is used in the present work.

Algorithm 3 Extraction the watermark

1. Decompose the watermarked image X' in non-overlapping blocks of size K pixels;
2. For each vector x'_i, execute the nearest codeword search to obtain a nearest codevector (w_j) in the codebook W;
3. The hidden watermark bit is the value that is contained in the corresponding position j of the key C;
4. Repeat steps 2-3 to extract every bit of the watermark.

4 Codebook partition with GA, PSO and EFA

4.1 Genetic Algorithm

The Genetic Algorithm Goldberg (1989); Katoch et al. (2021) is part of a class of algorithms based on Darwin's theory, in which the organisms best adapted to the environment have a greater chance of survival than the less adapted. It is an optimization algorithm that uses the concepts of evolutionary selection, mutation and recombination (crossover) as a search technique.

In the GA, the population of possible solutions, also called individuals, is composed of chromosomes representing a gene sequence. Let us assume, for example, each candidate solution is composed of a binary sequence and every element of this sequence values (0s and 1s) is called gene.

After the initialization process, each chromosome is evaluated. This value of the evaluation is known as fitness. The fitness assigned represents the adaptability of the individual on each generation of the population. The individuals with more adaptability have more chance to go to a new generation with the operator of selection. New individuals are generated by using the operator of the recombination. The diversity of the chromosomes emerges from the mutation operator Wazirali et al. (2019). Fig. 3 shows the steps of Genetic Algorithm.

The recombination stage is the creation of a new chromosome through the recombination of parent's genes, done according to a probability of crossing.

The mutation occurs according to a probability of mutation. In discrete problems, the mutation will be represented by the change in the value of the gene, i.e., if the value is 0, after the mutation it will change to the value 1, and if the value is 1, it will change to 0.

The Algorithm 4 is the GA Goldberg (1989) in the process of partitioning the original codebook into two sub-codebooks.

4.2 Particle Swarm Optimization

The Particle Swarm Optimization Kennedy and Eberhart (1995); Gad (2022) is an intelligence swarm algorithm

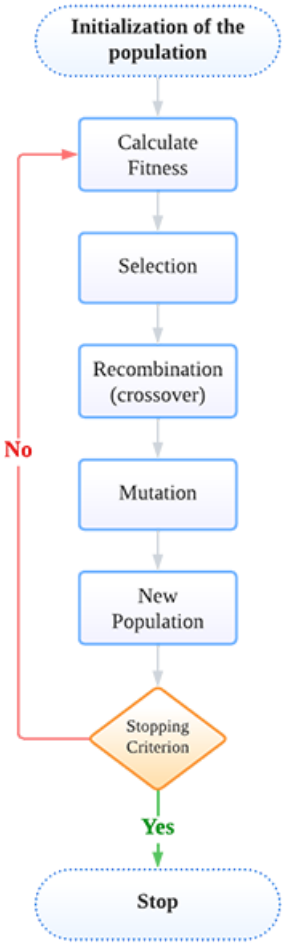

Figure 3: GA procedure.

Algorithm 4 Genetic codebook partition (GCP)

1. Generate a number of user-keys as chromosomes with dimensions (genes) of size C, with bits 0 and 1 randomly;
2. Insert the watermark in the original image using the sub-codebooks G_0 and G_1 partitioned by the user-key of the current chromosome;
3. Calculate the fitness value according to the given function (PSNR value obtained from the image reconstructed by the Step 2);
4. Select adapted individuals from a Selection Rate with the best fitness scores;
5. Output the best chromosome selected and terminate the training procedure if the considered iteration is met. Otherwise, keep executing the following steps;
6. Create new genes for next generation according to the pre-defined Crossover Rate;
7. Mutate the new chromosomes according to the Mutation Rate;
8. Go to Step 2.

inspired by the social behavior of a flock of birds that is based on collective communication among individuals searching for the solution by means of a global and local orientation Engelbrecht (2006).

Performing an analogy, the flock of birds (swarm) corresponds to possible solutions of the problem, the area overflown by the birds corresponds to the search space for each bird (particle). Each particle has coordinates (position and velocity vector) that will guide the bird in search of food, *i.e.* the best solution, influenced by the best known position of bird (PBEST) and the best position known by the band (GBEST).

Let $p_i(t)$ be the current position of particle i in the search space and time t. The new position is the sum of the current speed $v_i(t)$ with the current position, i.e.,

$$p_i(t+1) = p_i(t) + v_i(t). \tag{2}$$

The algorithm optimization is driven by the velocity vector, defined by

$$\begin{aligned} v_{ij}(t+1) &= IW \times v_{ij}(t) \\ &+ factor1 \times r1_j(t) \times [PBEST - p_{ij}(t)] \\ &+ factor2 \times r2_j(t) \times [GBEST - p_{ij}(t)], \end{aligned} \tag{3}$$

in which $v_{ij}(t)$ is the velocity of particle i in dimension j, $factor1$ and $factor2$ are positive constants used to scale the contribution of cognitive factor (PBEST) and social factor (GBEST), respectively. The terms $r1_j(t)$ and $r2_j(t)$ are random values between 0 and 1 obtained from a uniform distribution. The term IW (Inertia Weight) Shi and Eberhart (1998) is a positive constant to balance local and global search of the particles.

The Algorithm 5 is the PSO in the process of partitioning the original codebook into two sub-codebooks, i.e., creating the key that will be used in the watermark technique.

Algorithm 5 Proposed PSO codebook partition (PSOCP)

1. Initialize N particles (keys). Each particle is initialized with random values for its position and velocity vectors.
2. For each particle, insert the watermark in the original image using the sub-codebooks G_0 and G_1 partitioned by the user-key of the current particle, calculate the fitness (PSNR value obtained from the output image described in Algorithm 2) and evaluate the PBEST. Each particle is initialized with random values for its position and velocity vectors.
3. Evaluate GBEST;
4. If the number of iterations is met, then stop. Otherwise, update the particles velocity and particles position with Eq. (2);
5. Go to step 2.

4.3 Fireworks Algorithm

Fireworks Algorithm (FA) is a swarm intelligence algorithm inspired by watching fireworks explosions Tan and Zhu (2010). When the fireworks are launched, a set of sparks fill the place around the explosion. Thus, the explosion area limited by the amplitude can be seen as the search space and the sparks the possible solutions.

In the FA, for each generated explosion, first, locations Ln are chosen, where Fn fireworks will be launched. After the explosion, the sparks are scattered around the region and their positions are obtained and evaluated. When termination criterion is met for the problem, the algorithm ends. Otherwise, new explosions of the fireworks are set off from positions of sparks, generating a new sequence of the sparks.

FA is based on two behaviors in fireworks: the amount of generated sparks and their position around the explosion. The fireworks that have many sparks that are well spread around the explosion are considered good fireworks, because they create beautiful designs. The fireworks that have few sparks and few spread are considered bad fireworks.

From the point of view of a search algorithm, good sparks detonate fireworks located in one promising region. Some sparks should be used to scan the region, but they are not far from the explosion. The fireworks which are considered bad are far from the potential regions and with large amplitude and fewer sparks compared to good fireworks, where the goal is to use a small radius with more sparks, as we can see in Fig. 4 where the best solution is located at $(X, Y) = (0, 0)$.

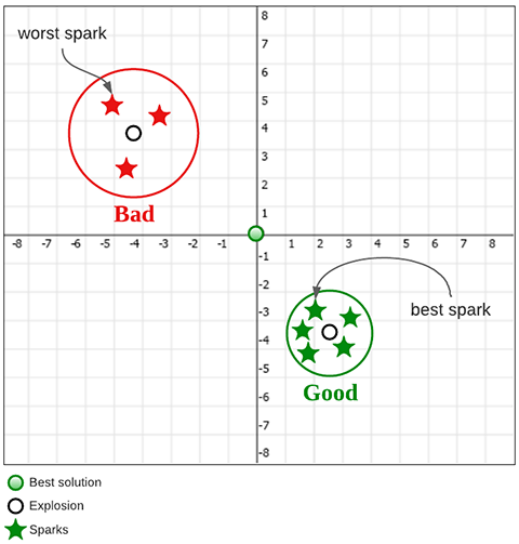

Figure 4: Example of good and bad fireworks.

Suppose FA is used to solve a minimization problem described by

$$Minimize\, f(x) \in \mathbb{R}, x_{min} \leq x \leq x_{max}, \tag{4}$$

in which the objective function is $f(x)$, and the search space locations are denoted by $x = x_1, x_2, \ldots, x_d$ and the limits of the search space are x_{max} and x_{min}. The number of sparks generated s for each of fireworks x_i is given by

$$s_i = m \times \frac{y_{max} - f(x_i) + \epsilon}{\sum_{i=1}^{n}(y_{max} - f(x_i)) + \epsilon}, \tag{5}$$

in which m is a parameter for controlling the number of sparks by the firework, ϵ is a constant used to avoid division by zero and y_{max} is the worst (highest in minimum problems) objective function value among the fireworks.

For good fireworks, the number of sparks must be high, but its amplitude should be small. The size of the amplitude is defined by

$$A_i = \hat{A} \times \frac{f(x_i) - y_{min} + \epsilon}{\sum_{i=1}^{n}(f(x_i) - y_{min}) + \epsilon}, \tag{6}$$

where $\hat{A}$ is the maximum size that amplitude can assume and $y_{min} = min(f(x_i))(i = 1, 2, \ldots, n)$ the best (smallest in minimization problems) value of the objective function found so far among the fireworks. After the explosion, the sparks are impacted and are scattered in the sky (search space). The direction z in which they move is random. This number is achieved as follows

$$z = round(d \times Rand(0, 1)), \tag{7}$$

where d is the dimensionality of the location x and $Rand(0, 1)$ is a value obtained from a uniform distribution in the interval [0,1]. The process to obtain the spark position is described by Algorithm 6.

Algorithm 6 Obtain the location of sparks

1. Start the positions of sparks: $\hat{x} = x_i$;
2. Select Z dimensions in a random form of $\hat{x}$;
3. Calculate the displacement of the impact: $h = A_i \times Rand(-1, 1)$;
4. For each dimension $\hat{x}_k^j$ preselected of $\hat{x}_j$ do:
   - **i.** $\hat{x}_k^j = \hat{x}_k^j + h$;
   - **ii.** If $\hat{x}_k^j < x_k^{min}$ or $\hat{x}_k^j > x_k^{max}$ then map $\hat{x}_k^j$ to the potential space $\hat{x}_k^j = x_k^{min} + |\hat{x}_k^j| \% (x_k^{max} - x_k^{min})$.

To maintain the diversity among sparks, mutant sparks are generated, or their displacements are calculated from a Gaussian distribution (new position is equal to generated displacement number plus current position). The last step of the algorithm is to choose the positions of the sparks that will be used for the launch for new fireworks. The best position will always be part of the new population and other $n - 1$ positions are chosen from a probability that depends on its distance to other sparks. The most common way to compute the distance between two sparks is given by

$$R(x_i) = \sum_{j \in k} d(x_i, x_j) = \sum_{j \in k} ||x_i - x_j||, \tag{8}$$

in which k is the set of all positions. The probability of choice is given by

$$p(x_i) = \frac{R(x_i)}{\sum_{j \in k} R(x_j)}. \tag{9}$$

4.4 Enhanced Fireworks Algorithm

Enhanced Fireworks Algorithm (EFA) Zheng et al. (2013); Yue et al. (2020) uses five modifications to the classical version of the algorithm: new minimum amplitude check, new operator for generating explosion sparks, new mapping operator, new explosion operator of the mutant sparks and new selection operator.

As described in the previous section, the fireworks with the best fitness will have a high number of sparks and a low amplitude. If the amplitude of the explosion is close to zero, this will cause some sparks to take almost the same position, losing diversity in the swarm. To correct this problem a minimum range checker based on the algorithm's progress is added (Eq. (10)). At the beginning of the search, the lower limit range of A_{min} is high, but with the increase in the number of iterations of the algorithm, the value of A_{min} is diminished. For each dimension K, the explosion amplitude A_i^k is limited as follows:

$$A_i^k = \begin{cases} A_{min}^k \; if \; A_i^k < A_{min}^k \\ A_i^k, \; otherwise \end{cases}. \tag{10}$$

In Algorithm 6, it is observed that the displacement of the sparks is the same for all positions. Thus, in order to better exploit the search space, a new sparks generation process is proposed to positions. The change consists in using an offset ΔX, which is different for each shift position. Let $\hat{X}_l^k$ be the size of the sparks that suffer displacement. It is calculated as follows:

$$\Delta X^k = A_i \times rand(-1, 1) \tag{11}$$

$$\hat{X}_l^k = \hat{X}_l^k + \Delta X^k. \tag{12}$$

In the classic version of the Fireworks algorithm, when a new spark location exceeds a range of values of the search space of a dimension k $[X_{min}^k, X_{max}^k]$, this new spark is mapped to a new value. At several points, the exceeded amount is not far from the established limit, and the new value generated causes the spark to take near feasible space Tan and Zhu (2010). To solve this problem, the spark is mapped by

$$\bar{X}_i^k = X_{min}^k + rand \times (X_{max}^k - X_{min}^k). \quad (13)$$

To improve the location of these sparks (perform better exploitation of the search space), the operator of mutant sparks is now calculated as $X_i^k = X_i^k + (X_B^k - X_i^k) \times e$, in which X_B is the current location of the best fireworks/sparks found thus far and e is a value obtained from a normal distribution with variance equal to 1 according to Zheng et al. (2013).

The EFA selection operator is based on a strategy involving the distance between the solutions in sparsely populated regions. This strategy has a large computational cost, with great impact on the algorithm execution time. The EFA uses the Elitism-Random Selection selection method, i.e., the new population of fireworks is comprised by the best and worst solution found so far and $n - 2$ random fireworks. As a result, the running time is much lower.

Algorithm 7 presents the EFA for partitioning of the original codebook for key generation.

Algorithm 7 Proposed EFA codebook partition (EFACP)

1. Initialize the F fireworks as keys with L locations represented by bits 0 and 1 with dimension K;
2. Calculate the amplitude, number of regular sparks (s) and generate the regular sparks of each firework;
3. Generate mutant sparks;
4. For each dimension (K), randomly select a firework and generate sparks;
5. Evaluate each spark created using the fitness function (PSNR value);
6. Select the best and worst spark and s - 2 sparks randomly for new generation fireworks;
7. If the number of iterations is met, return the position of the best spark found. Otherwise, go to step 2.

5 Methodology and results

The methodology of this paper consisted of using the technique introduced in Wang et al. (2007). In the present work, we investigate the use of the computational intelligence algorithms PSO and EFA for the codebook partitioning. Each algorithm (GCP, PSOCP and EFACP) was executed 30 times and the algorithm LBG was used for the codebook design. The settings of the Genetic Algorithm were obtained from Wang et al. (2007) and are presented in Table 1. The configuration settings of the LBG, PSO, and Enhanced Fireworks algorithms were chosen according to their literature and are presented respectively in Table 2, Table 3 and Table 4.

The simulations were performed using three images 512×512 pixels with 256 gray levels (8 bpp) shown in Fig. 5. The quantized images Elaine, Peppers, and Man obtained the PSNR values of 31.49 dB, 31.27 dB and 26.49 dB without the insertion of the watermark (that is, reconstruction by using the entire codebook), respectively. The image Rose with 128×128 pixels and 1 bit per pixel was used as the watermark, shown in Fig. 6. As the key that makes the codebook partition is a binary key (0s and 1s), the algorithm PSO was used in its binary version and, in the same way, the Fireworks Algorithm.

Table 1: Genetic Algorithm original settings used by the authors in Wang et al. (2007).

Population size	10
Number of iterations	1000
Selection rate	100%
Crossover rate	50%
Mutation Rate	0.1%
Selection operator	Roulette Wheel

Table 2: Settings of codebook designed by the algorithm LBG.

Distortion error threshold	10^{-4}
Codebook size	256
Dimension K	16 (blocks 4×4)

Table 3: PSO settings.

Number of particles	10
Number of iterations	1000
IW	1
$factor1$	2
$factor2$	2

Table 4: Enhanced Fireworks Algorithm settings.

Number of iterations	200
Number of fireworks	5
Number of regular sparks	25
Number of mutant sparks	5
Maximum value of generated sparks	0.8
Minimum value of generated sparks	0.04
Maximum amplitude	70

To assess the performance of the computational intelligence algorithms tested, the PSNR of the images and the Bit Correct Ratio (BCR) are used. The BCR is calculated as

$$\text{BCR(Y,Y')} = \left(1 - \frac{\sum_{i=1}^{L_W} |y_i - y_i'|}{L_W} \right) \times 100\%, \quad (14)$$

in which Y is the original watermark, Y' is the extracted watermark, L_W is the size of the watermark, y_i and y_i' are the ith bits of Y and Y', respectively. A variety of attacks was tested: JPEG compression with quality factor of 40%, 60% and 80%, the filters mean and median, 25% Cropping (third quadrant), Gaussian and Salt and Pepper noise with σ (the standard deviation of the Gaussian distribution used to generate the noise) and density (the

proportion of pixels in an image that are affected by the salt and pepper noise) of 0.001 and 0.005, respectively, Poisson and shifting one line downward. Thus, high values of BCR indicate high similarity between the original watermark and the extracted watermark.

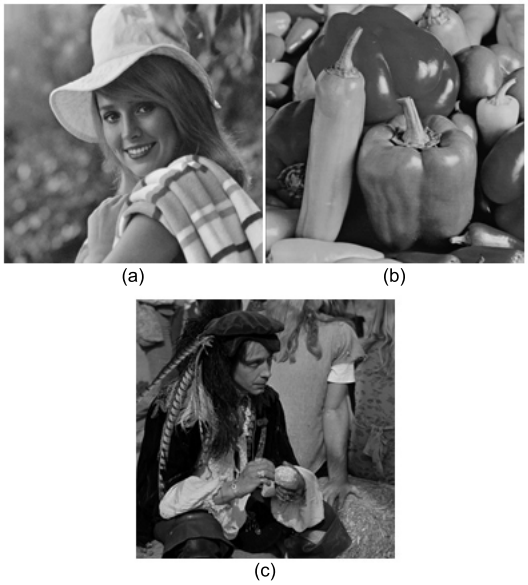

Figure 5: Images used in the simulations. (a) Elaine, (b) Peppers and (d) Man.

Figure 6: The watermark.

The algorithms were implemented using the C# programming language with the VS Code in a computer with AMD Ryzen 5600x, clock of 3.70 GHz, 32 GB of DD4 RAM and the operating system Windows 10.

5.1 Quality of watermarked images

The following results are obtained from the algorithms GCP, PSOCP, EFACP used for partitioning the original codebook into sub-codebooks. Table 5 shows the results of best PSNR and average PSNR of the watermarked images and Table 6, Table 7 and Table 8 present the BCR results obtained from the watermark extracted after the attacks.

In Table 5, the highest PSNR value achieved by the PSO algorithm for image Peppers was 30.03 while the PSNR of the quantized image was 31.49 dB. The difference of 1.46 dB is perceptually observed, as can be seen in Fig. 7.

Table 5: Best and average PSNR obtained in simulations.

IMAGE	PSNR	GA	PSO	EFA
Elaine	Best	30.58	**30.61**	30.59
	Average	30.55	**30.60**	30.58
Man	Best	25.58	**25.60**	25.59
	Average	25.55	**25.59**	25.57
Peppers	Best	30.00	**30.03**	30.01
	Average	29.95	**30.01**	29.99

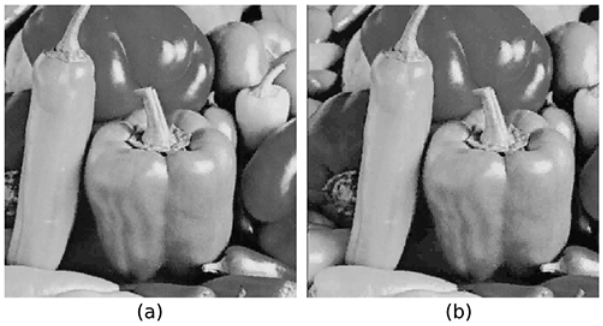

Figure 7: (a) Quantized image and (b) Watermarked image.

As observed in Table 6 to Table 8, attacks such as Poisson and Gaussian noise tend to yield more pronounced impacts in the watermark. The sensitivity to changes varies among images. When applying JPEG compression with a quality factor of 40% for GA, the Pepper image results in a BCR of 85.71%. In contrast, the Man and Elaine images yields BCRs of 93.50% and 81.01%, respectively.

Table 6: BCR values of the watermark extracted from Man after attacks.

Attacks	BCR (%) GA	PSO	EFA
JPEG, QF=40%	93.50	93.30	93.66
JPEG, QF=60%	96.25	96.75	96.78
JPEG, QF=80%	98.50	98.61	98.24
Mean filtering	74.50	74.11	74.73
Median filtering	88.39	88.24	88.52
Cropping, 25%	80.26	94.74	80.26
Shifting one line	72.73	72.38	73.68
Gaussian noise (0.001)	56.36	56.23	56.42
Gaussian noise (0.005)	56.24	55.98	55.71
Poisson	76.50	76.01	76.50
Salt and Pepper (0.001)	99.35	99.29	99.34
Salt and Pepper Salt (0.005)	96.53	96.44	96.51

From Fig. 6, the original watermark Rose, we can identify that the region affected by the Cropping attack 25% consists of more black pixels than white pixels. Thus,

the result of 94.74% for BCR, i.e. greater than other results (80.26%), is an expected result because the watermark extracted has more black pixels in its third quadrant as can be seen in Fig. 8.

Fig. 9 shows the result on the watermarks extracted for some common attacks. For all the attacks presented in Fig. 9 for image Peppers, the shifting one line downward was the one that most degraded the extracted watermark, with a corresponding value 70.36% of BCR. However, it is still possible to see the rose. The numerical results of Table 7 are in consonance with what one observes in Fig. 9. Indeed, among the aforementioned attacks, the lowest BCR is associated with shifting one line downward.

Table 7: BCR values of the watermark extracted from Peppers after attacks.

Attacks	BCR (%) GA	PSO	EFA
JPEG, QF=40%	85.71	86.68	87.17
JPEG, QF=60%	94.73	94.95	94.42
JPEG, QF=80%	98.08	98.37	98.42
Mean filtering	81.15	81.57	81.37
Median filtering	94.53	95.02	94.65
Cropping, 25%	94.74	94.74	94.74
Shifting one line	70.36	69.79	69.24
Gaussian noise (0.001)	56.78	56.02	56.54
Gaussian noise (0.005)	56.67	56.87	57.23
Poisson	74.04	74.33	74.66
Salt and Pepper (0.001)	99.36	99.23	99.22
Salt and Pepper Salt (0.005)	96.63	96.42	96.55

Table 8: BCR values of the watermark extracted from Elaine after attacks.

Attacks	BCR(%) GA	PSO	EFA
JPEG, QF=40%	81.01	78.33	80.52
JPEG, QF=60%	88.71	86.10	87.37
JPEG, QF=80%	95.29	93.50	93.41
Mean filtering	74.41	72.20	72.92
Median filtering	81.97	79.55	79.31
Cropping, 25%	94.74	94.74	94.74
Shifting one line	61.05	59.14	61.04
Gaussian noise (0.001)	54.69	55.19	54.27
Gaussian noise (0.005)	54.51	55.35	54.65
Poisson	69.96	70.23	69.33
Salt and Pepper (0.001)	99.33	99.32	99.38
Salt and Pepper Salt (0.005)	96.26	96.19	96.27

The lowest BCR value obtained among all images tested is 54.27% (Table 8) for the Elaine image obtained by EFA after Gaussian noise attack. Fig. 10 shows the watermarked image before and after the attack as well as the watermark extracted, which is very degraded.

We can also observe that the salt and pepper attack with a small density (0.001) leads to high BCR values, almost 100%. However, it is important to note that the extracted watermark still shows signs of being affected by the attack with minor distortions. For higher density value (0.005),

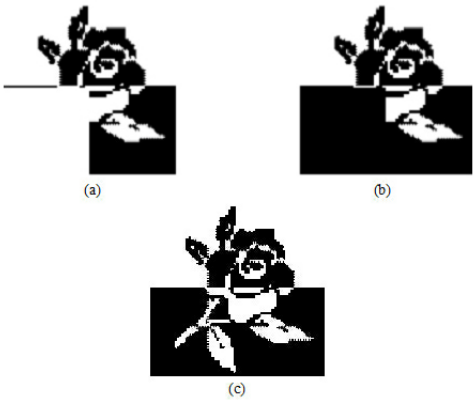

Figure 8: Watermark extracted with BCR value (a) 80.26%, (b) 94.74% and (c) the original watermark.

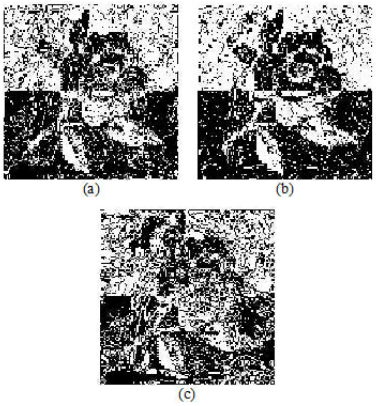

Figure 9: Watermark extracted from image Peppers for a key generated by the algorithm GA for the attacks (a) mean, (b) JPEG, QF = 40% and (c) shifting one line downward.

the impact of the attack becomes more pronounced, as shown in Fig. 11.

5.2 Execution time of the algorithms GA, PSO and EFA

The algorithms were limited to 5000 calls to the fitness function and the Partial Distortion Search (PDS) Bei and Gray (1985) technique was used in the Nearest Neighbor Search to all algorithms. A version of PSO without PDS, i.e, full search (PSO FS) was also evaluated. Table 9 presents the lowest time (best) and average time (average) obtained.

The comparison of the algorithms in terms of PSNR (Table 10) and total running time (Table 9) revealed that PSO achieved slightly higher PSNR values compared to GA and EFA. Additionally, PSO demonstrated superior computational efficiency, as it requires running times shorter than those of GA and EFA.

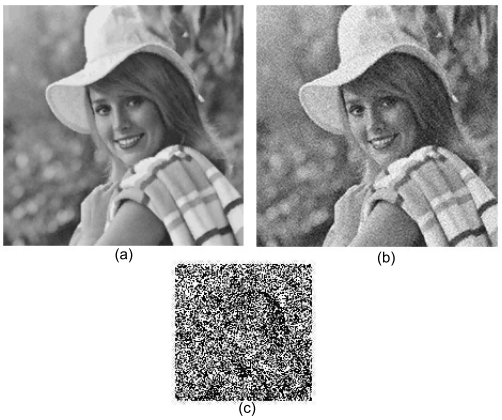

Figure 10: (a) Watermarked Elaine image, (b) applied Gaussian noise with σ = 0.001 attack to the watermarked image of Elaine and (c) the watermark extracted after attack.

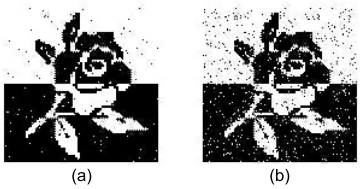

Figure 11: Watermark extracted from image Elaine for Salt and Pepper attack with σ (a) 0.001 and (b) 0.005.

Table 9: Total running time in seconds for the algorithms tested.

IMAGE	TIME	GA	PSO	PSO FS	EFA
Elaine	Best	266.58	253.79	307.41	339.02
	Average	268.93	255.63	308.22	346.75
Man	Best	281.73	266.81	306.02	257.92
	Average	283.45	268.72	306.88	363.83
Peppers	Best	235.08	224.31	306.47	300.73
	Average	237.97	226.03	307.20	307.08

Table 10: Results for PSNR values for Elaine image with the number of calls to the fitness function fixed at 5000.

PSNR	GA	PSO	EFA
Best	30.60	30.61	30.58
Average	30.58	30.59	30.56

6 Conclusion

This paper presents the use of PSO and EFA algorithms as alternatives to the Genetic Algorithm in the codebook partition step in the watermarking technique described in Wang et al. (2007). A robustness analysis of the watermarking technique was carried out.

The results demonstrated that PSO not only provided comparable or superior image quality (in terms of PSNR) but also significantly reduced execution time, especially due to the use of the Partial Distortion Search (PDS) method. On the other hand, EFA did not yield substantial improvements in either image quality or resistance to attacks, and presented a higher computational cost.

The robustness of the watermarking was validated against several attacks, including JPEG compression, filtering, cropping, noise addition, and geometric transformation. Among the tested methods, PSO consistently offered a favorable trade-off between robustness and efficiency.

These findings suggest that PSO is a promising candidate for watermark embedding in compressed image domains. Future research could explore other optimization strategies or the integration of deep learning to improve the robustness and adaptability of watermarking techniques.

Acknowledgments

The authors would like to thank Fundação de Amparo a Ciência e Tecnologia de Pernambuco (FACEPE) and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the financial support to the work. This study was partially funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020-UIDP/50008/2020

References

Amrit, P. and Singh, A. K. (2022). Survey on watermarking methods in the artificial intelligence domain and beyond, *Computer Communications* **188**: 52–65. http://dx.doi.org/10.1016/j.comcom.2022.02.023.

Azevedo, C. R., Azevedo, F. E., Lopes, W. T. and Madeiro, F. (2009). Terrain-based memetic algorithms for vector quantizer design, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, Springer, pp. 197–211. http://dx.doi.org/10.1007/978-3-642-03211-0_17.

Bei, C.-D. and Gray, R. (1985). An improvement of the minimum distortion encoding algorithm for vector quantization, *IEEE Transactions on Communications* **33**(10): 1132–1133. http://dx.doi.org/10.1109/TCOM.1985.1096214.

Bispo Jr, E., Azevedo, C., Lopes, W., Alencar, M. and Madeiro, F. (2010). Methods to accelerate a competitive learning algorithm applied to VQ codebook desing, *Trends in Computational and Applied Mathematics* **11**(3): 193–203. http://dx.doi.org/10.5540/tema.2010.011.03.0193.

Camacho-Gonzalez, F. D., Lima-López, D., Zapotecas-Martínez, S. and Altamirano-Robles, L. (2025). Vector quantization-driven image compression through multi-objective evolutionary algorithms, *Expert Systems with Applications* **261**: 125512. https://doi.org/10.1016/j.eswa.2024.125512.

Engelbrecht, A. P. (2006). *Fundamentals of computational swarm intelligence*, John Wiley & Sons, Inc. http://dx.doi.org/10.5555/1199518.

Evsutin, O., Melman, A. and Meshcheryakov, R. (2020). Digital steganography and watermarking for digital images: A review of current research directions, *IEEE Access* **8**: 166589–166611. http://dx.doi.org/10.1109/ACCESS.2020.3022779.

Fei, J., Xia, Z., Tondi, B. and Barni, M. (2022). Supervised GAN watermarking for intellectual property protection, *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, pp. 1–6. http://dx.doi.org/10.1109/WIFS55849.2022.9975409.

Fonseca, C., Ferreira, F. A. and Madeiro, F. (2018). Vector quantization codebook design based on fish school search algorithm, *Applied Soft Computing* **73**: 958–968. http://dx.doi.org/10.1016/j.asoc.2018.09.025.

Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review, *Archives of Computational Methods in Engineering* **29**(5): 2531–2561. https://doi.org/10.1007/s11831-021-09694-4. **URL:** *https://doi.org/10.1007/s11831-021-09694-4*

Gersho, A. and Gray, R. M. (2012). *Vector quantization and signal compression*, Vol. 159, Springer Science & Business Media. http://dx.doi.org/10.1007/978-1-4615-3626-0.

Goldberg, D. (1989). Genetic algorithms in search, optimization, and machine learning. choice rev, *Online* **27**(02): 27–0936. http://dx.doi.org/10.5860/choice.27-0936.

Gray, R. (1984). Vector quantization, *IEEE ASSP Magazine* **1**(2): 4–29. http://dx.doi.org/10.1109/MASSP.1984.1162229.

Issa, M. (2018). Digital image watermarking performance improvement using bio-inspired algorithms, *Advances in Soft Computing and Machine Learning in Image Processing* pp. 683–698. http://dx.doi.org/0.1007/978-3-319-63754-9_30.

Jiang, W., Liu, P. and Wen, F. (2017). An improved vector quantization method using deep neural network, *AEU-International Journal of Electronics and Communications* **72**: 178–183. http://dx.doi.org/10.1016/j.aeue.2016.12.002.

Joshi, A. S., Kulkarni, O., Kakandikar, G. M. and Nandedkar, V. M. (2017). Cuckoo search optimization-a review, *Materials Today: Proceedings* **4**(8): 7262–7269. http://dx.doi.org/10.1016/j.matpr.2017.07.055.

Katoch, S., Chauhan, S. S. and Kumar, V. (2021). A review on genetic algorithm: past, present, and future, *Multimedia Tools and Applications* **80**(5): 8091–8126. https://doi.org/10.1007/s11042-020-10139-6. **URL:** *https://doi.org/10.1007/s11042-020-10139-6*

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4, pp. 1942–1948. http://dx.doi.org/10.1109/ICNN.1995.488968.

Linde, Y., Buzo, A. and Gray, R. (1980). An algorithm for vector quantizer design, *IEEE Transactions on Communications* **28**(1): 84–95. http://dx.doi.org/10.1109/TCOM.1980.1094577.

Mata, E., Bandeira, S., de Mattos Neto, P., Lopes, W. and Madeiro, F. (2016). Accelerating families of fuzzy k-means algorithms for vector quantization codebook design, *Sensors* **16**(11): 1963. http://dx.doi.org/10.3390/s16111963.

Roy, R., Ahmed, T. and Changder, S. (2018). Watermarking through image geometry change tracking, *Visual Informatics* **2**(2): 125–135. http://dx.doi.org/10.1016/j.visinf.2018.03.001.

Sanivarapu, P. V., Rajesh, K. N., Hosny, K. M. and Fouda, M. M. (2022). Digital watermarking system for copyright protection and authentication of images using cryptographic techniques, *Applied Sciences* **12**(17): 8724. http://dx.doi.org/10.3390/app12178724.

Severo, V., Ferreira, F. B., Spencer, R., Nascimento, A. and Madeiro, F. (2024). On the initialization of swarm intelligence algorithms for vector quantization codebook design, *Sensors* **24**(8): 2606. http://dx.doi.org/10.3390/s24082606.

Severo, V., Leitão, H., Lima, J., Lopes, W. and Madeiro, F. (2016). Modified firefly algorithm applied to image vector quantisation codebook design, Vol. 7, Inderscience Publishers (IEL), pp. 202–213. http://dx.doi.org/10.1504/IJICA.2016.080859.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer, *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, IEEE, pp. 69–73. http://dx.doi.org/10.1109/ICEC.1998.699146.

Su, G.-D., Chang, C.-C. and Lin, C.-C. (2020). Effective self-recovery and tampering localization fragile watermarking for medical images, *IEEE Access* **8**: 160840–160857. http://dx.doi.org/10.1109/ACCESS.2020.3019832.

Sy, N. C., Kha, H. H. and Hoang, N. M. (2020). An efficient robust blind watermarking method based on convolution neural networks in wavelet transform domain, *International Journal of Machine Learning and Computing* **10**: 675–684. http://dx.doi.org/10.18178/ijmlc.2020.10.5.990.

Tan, Y. and Zhu, Y. (2010). Fireworks algorithm for optimization, *Advances in Swarm Intelligence: First International Conference, ICSI 2010, Beijing, China, June 12-15, 2010, Proceedings, Part I 1*, Springer, pp. 355–364. http://dx.doi.org/10.1007/978-3-642-13495-1_44.

Wang, F.-H., Jain, L. C. and Pan, J.-S. (2007). VQ-based watermarking scheme with genetic codebook partition, *Journal of Network and Computer Applications* **30**(1): 4–23. http://dx.doi.org/10.1016/j.jnca.2005.08.002.

Wang, Z., Bovik, A. C., Sheikh, H. R. and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity, *IEEE Transactions on Image Processing* **13**(4): 600–612. http://dx.doi.org/10.1109/TIP.2003.819861.

Wang, Z.-X., Sha, K.-Y. and Gao, X.-L. (2022). Digital watermarking technology based on ldpc code and chaotic sequence, *IEEE Access* **10**: 38785–38792. http://dx.doi.org/10.1109/ACCESS.2022.3166475.

Wazirali, R., Alasmary, W., Mahmoud, M. M. and Alhindi, A. (2019). An optimized steganography hiding capacity and imperceptibly using genetic algorithms, *IEEE Access* **7**: 133496–133508. http://dx.doi.org/10.1109/ACCESS.2019.2941440.

Yue, Z., Zhang, S. and Xiao, W. (2020). A novel hybrid algorithm based on grey wolf optimizer and fireworks algorithm, *Sensors* **20**(7). https://doi.org/10.3390/s20072147.

Zheng, S., Janecek, A. and Tan, Y. (2013). Enhanced fireworks algorithm, *2013 IEEE congress on Evolutionary Computation*, IEEE, pp. 2069–2077. http://dx.doi.org/10.1109/CEC.2013.6557813.