Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

A stable and fast PSO algorithm guided by SPSA for vector quantization-based image compression

Vector nicemleme tabanlı görüntü sıkıştırma için EPSY ile yönlendirilen kararlı ve hızlı bir PSO algoritması

İlker KILIÇ[1*], Haldun SARNEL[1]

[1]Department of Electrical and Electronics Engineering, Faculty of Engineering and Natural Sciences, Manisa Celal Bayar University Manisa, Türkiye.

ilker.kilic@cbu.edu.tr, haldun.sarnel@cbu.edu.tr

Received/Geliş Tarihi: 17.10.2024 Revision/Düzeltme Tarihi: 25.02.2025 doi: 10.5505/pajes.2025.78006
Accepted/Kabul Tarihi: 24.04.2025 Research Article/Araştırma Makalesi

Abstract

Image compression plays a crucial role in reducing storage requirements and improving transmission efficiency. The effectiveness of lossy image compression using vector quantization (VQ) heavily depends on the quality of codebook generation, which is inherently an optimization problem. In this paper, a coupled hybrid algorithm integrating Simultaneous Perturbation Stochastic Approximation (SPSA) into Particle Swarm Optimization (PSO) is proposed to enhance both the convergence speed and codebook quality in vector quantization. The novel SPSA-FPSO algorithm, by generating multiple alternative codebooks at each iteration and selecting the best, successfully avoids local minima and achieves faster convergence. Experimental results, conducted on standard gray-level images of various contrast levels, demonstrate that the proposed SPSA-FPSO algorithm outperforms both basic PSO and SPSA algorithms in terms of lower mean square error (MSE) and higher convergence speeds, establishing its superiority for VQ-based image compression tasks. This superiority is also shown to be valid when compared to other metaheuristic algorithms.

Keywords: Image compression, Metaheuristic algorithms, Vector Quantization, Codebook generation, PSO, SPSA

Öz

Görüntü sıkıştırma, depolama gereksinimlerini azaltmak ve iletim verimliliğini artırmak açısından büyük bir öneme sahiptir. Vektör nicemleme (VN) tabanlı kayıplı görüntü sıkıştırmanın başarısı, esasen bir optimizasyon problemi olan kod tablosu üretiminin kalitesine bağlıdır. Bu makalede, hem algoritmanın yakınsama hızını hem de VN kod tablosunun kalitesini artırmak için Eşzamanlı Pertürbasyon Stokastik Yaklaşımı (EPSY) tekniğini Parçacık Sürü Optimizasyonu (PSO) ile bütünleştiren hibrit bir algoritma önerilmektedir. Önerilen EPSY-HPSO algoritması, her iterasyonda birden fazla alternatif kod kitabı üreterek en iyisini seçmekte ve yerel minimum noktalarından kaçınarak daha hızlı bir yakınsama sağlamaktadır. Farklı kontrast seviyelerine sahip standart gri seviye görüntüler üzerinde gerçekleştirilen deneysel sonuçlar, EPSY-HPSO algoritmasının hem ortalama kare hata (OKH) değerlerini düşürme hem de daha yüksek yakınsama hızları açısından klasik PSO ve EPSY algoritmalarından daha başarılı olduğunu göstererek VN tabanlı görüntü sıkıştırmadaki üstünlüğünü kanıtlamaktadır. Bu üstünlüğün diğer metasezgisel algoritmalarla karşılaştırıldığında da geçerli olduğu gösterilmektedir.

Anahtar kelimeler: Görüntü sıkıştırma, Metasezgisel algoritmalar, Vektör Nicemleme, Kod tablosu üretimi, PSO, EPSY

1 Introduction

Digital image compression is a process which aims at reducing the size of a digital image for both storage and transmission purposes. Compressed images require less information storage capacity on storage devices, while allowing lower bandwidth and less time duration to transmit them over a communication channel. With the help of digital image compression algorithms, all digital devices using image data are today able to handle high-resolution large-size digital images in a more efficient manner.

Vector quantization (VQ) [1],[2] is a widely used quantization technique in lossy signal processing, information classification, pattern recognition and feature extraction. In lossy image compression, small image blocks in a whole image are grouped based on their similarity. The VQ algorithm reduces the image size by mapping blocks in an image into a set of vectors. In implementation, after the division of the image into small blocks, each block is converted to a vector. Similar vectors form a cluster centered at a codeword vector. All codeword vectors of the image form a list called codebook. All image vectors are assigned to the codeword, which is the most similar in the codebook, and represented by their addresses in the codebook instead of the vector itself. Therefore, a compression is obtained, and the image size is reduced. The codebook including a number of codewords must be estimated to represent the entire image as close to its original as possible.

Better codebook estimation is a crucial task for improving the performance and efficiency of the image compression algorithms. Usually, arithmetic coding and Huffman algorithm [3] are opted for a lossless image compression process. However, arithmetic coding necessitates extensive probability distribution tables to define symbol codes, and the frequency-based code definitions in Huffman coding significantly impact compression efficiency. Due to these limitations, alternative lossy compression techniques have become more popular and achieve higher compression ratios. In VQ-based compression methods, codebook estimation has a great effect on the performance [4]. Here, better grouping the image blocks directly decreases the error of the reconstructed image. Therefore, finding the optimal codebook can be viewed as an optimization problem. Codebook optimization increases the quality of the reconstructed image at the same compression ratios. The use of metaheuristics for codebook generation has

[*]Corresponding author/Yazışılan Yazar

become more popular recently. The goal of those optimization algorithms is to find better individuals for representing the optimum codebook. There are many metaheuristic methods in the literature presented to solve the codebook estimation problem in VQ. Some of them are genetic algorithm [5], cuckoo search algorithm [6], ant colony optimization [7], particle swarm optimization [8], firefly optimization algorithm [9], bat algorithm [10], lion optimization algorithm [11], flower pollination algorithm [12], whale optimization algorithm [13], and crow search algorithm [14]. There are also some metaheuristic methods enhanced by useful techniques to increase their performances for optimal codebook estimation such as improved differential evaluation algorithm [15], improved sine–cosine algorithm [16], improved adolescent identity search algorithm [17], smart fruit fly optimization algorithm [18] and Levy flight based bat algorithm [19]. A can finally be added to A new Some of these metaheuristic optimization algorithms use the K-Means technique to find initial codebooks before starting the main optimization process.

Simultaneous perturbation stochastic approximation (SPSA) is an optimization algorithm developed for difficult multivariate optimization problems [20]. The gradient approximation of an optimization problem is the primary benefit of the SPSA. The algorithm needs only two evaluations of the objective function per iteration to estimate the unknown parameters regardless of the dimension of the optimization problem. Therefore, the SPSA has a superior performance for estimating the gradient value without direct gradient information. This property makes it favourable to use in many optimization problems including signal and image processing, feedback control, statistical parameter estimation, and simulation-based optimization.

Particle swarm optimization (PSO) is a population (or swarm) based optimization algorithm using stochastic search. It was inspired by the social behavior of animals and relies on the assumption that social exchange of information among individuals could lead to evolutionary distinction. Each particle in the swarm represents an individual of the population. PSO is one of the best known and commonly used meta-heuristic algorithms. But it has two major drawbacks. First, the basic algorithm has a relatively slow progress towards optimal or near optimal solutions, especially when the problem domain has a search space of high dimension. Moreover, the position update strategy of the particles, that depends strongly on the global best (*gbest*) particle, has also adverse effects on the algorithm performance. In PSO, *gbest* inherently guides the rest of the swarm. Unfortunately, this causes the swarm to become more similar to this guiding particle, bringing with it some loss of diversity in the search space. It is exactly this phenomenon that increases the probability of premature convergence and getting trapped in local minima. To avoid these drawbacks and improve the overall performance of the PSO algorithm, many hybrid schemes are proposed in the literature. Some examples of such schemes are given as follows. Chen *et al*. [21] propose a local search method based on the conjugate gradient in combination with the PSO for the identification of nonlinear systems. A hybrid PSO and ant colony optimization [22] is proposed to solve the problem of designing truss structures. A quasi-Newton sequential quadratic programming method for local search is combined with the PSO algorithm in [23] to solve structural optimization problems. Cherki *et al.* [24] use a sequential combination of GA and PSO to solve the problem of the optimal power flow on electrical networks. Seyedpoora *et al.* [25] propose a preliminary optimization using SPSA followed by PSO. In their scheme, many copies of the result provided by SPSA are appended to the usual randomized initial swarm for the PSO algorithm. This process produces an efficient hybrid initial swarm. Therein it is shown that their hybrid algorithm performs well in solving structural optimization problems. In another study [26], combining PSO with gradient-based methods for optimizing convolutional neural networks is explored. Following some performance comparisons in their research, the results come out generally in favor of conventional gradient-based methods rather than the hybridization of PSO.

The hybrid algorithms given above combine two among many optimization algorithms in a way to apply them sequentially and are classified in a decoupled hybridization type. Alternatively, the PSO has been hybridized with several popular metaheuristics and mathematical solvers such as gradient-oriented schemes [27] in a coupled way. Hybridization with the genetic algorithm [28], the differential evolution algorithm [29], the harmony search algorithm [30], the sine–cosine algorithm [31], the gray wolf algorithm [32], the firefly algorithm [33], and finally both whale and differential evolution algorithms (multiple hybridization) [34] can be given as some other examples of the second type. It appears that more research focuses on the coupled hybridization because its designs significantly affect the performance of the base algorithms. As a result, its implementations have shown enhancements in the quality of the optimization results.

Another coupled hybridization of PSO to avoid its drawbacks is given by Kiranyaz *et al*. [35]. They propose two algorithms in which SPSA is embedded inside PSO to guide the swarm. They use a generic name *stochastic approximation driven* PSO (SAD PSO) for their algorithms. In their first algorithm, *gbest* is updated (or guided) only by SPSA with the ability for gradient estimation of the objective function while other particles are updated as usual at every PSO iteration. Their second algorithm utilizes the information of a special particle which is not in the swarm, so-called alternative global best particle. The position of this particle is computed by guiding *gbest* with SPSA. The decision is made based on a competition between the two. The winner replaces the particle *gbest* to provide better guidance to the swarm in the next iteration.

The use of metaheuristic methods for VQ-based image compression has been limited to the use of base algorithms alone. With a single base algorithm, the optimum codebook estimation process can get stuck at local optima, resulting in compressed images with worse visual quality than the optimum level. Even in cases where local optima can be escaped from, the number of iterations required to reach the global optimum is quite large, overshadowing the practical importance of the algorithm for codebook optimization. The motivation of this work is to improve the accuracy and speed performance of a metaheuristic method for the codebook estimation problem by using a hybridization approach. PSO was chosen as the metaheuristic base algorithm since it is a well-known optimization algorithm. The inspiration for developing our optimization algorithm is the second SAD PSO algorithm in [35]. Although their hybrid algorithm provided performance improvement over PSO, it can be further improved in terms of the quality of the guidance given by SPSA for both increasing diversity in the search space and speeding up the convergence. In this paper, we first propose an extended algorithm. The proposed algorithm computes multiple SPSA gradient approximations instead of one per iteration and selects the best one to find a better next position (codebook) for *gbest*. This extension can also be stated as using multiple alternative global particles instead of one in the previous algorithm. Additionally, the gain parameter a of the SPSA algorithm is increased dynamically through the iterations in the proposed algorithm. This novelty results in further performance increase. The proposed extended algorithm will be referred to as SPSA-*driven fast* PSO (SPSA-FPSO) from here on to emphasize the guidance role of SPSA in the basic PSO and the acceleration provided in the convergence speed. Second major contribution of this work is the presentation and discussion of the results obtained from the first-time application of this high performance algorithm to codebook estimation in VQ-based image compression. Unlike the PSO algorithm's frequent result of getting trapped in local optima, the proposed SPSA-FPSO algorithm is capable of escaping from local optima and always reaches or approximates the global optimum.

This paper is organized as follows. Section 2 introduces the K-Means algorithm for VQ, the basic PSO algorithm, and the SPSA algorithm. The proposed algorithm and its application to image compression based on VQ are presented in Section 3. The experimental results over mean square error (MSE) minimization between the reconstructed images and the originals are given in Section 4. Finally, Section 5 provides a summary of the concluding remarks.

2 Related methods

In this section, the K-Means algorithm for VQ that produces initial codebooks for all optimization algorithms, the PSO algorithm and the SPSA algorithm are explained, respectively, in detail.

2.1 K-Means algorithm for vector quantization

The K-Means algorithm is a plain and well-known clustering technique used in many application areas including Vector Quantization. The algorithm aims to partition the whole data into "k" Voronoi cells. K-Means algorithm can be used to cluster data vectors converted from small image blocks. It is one of the most favorite choices for generating codebooks in VQ due to its simplicity and fair image representation ability. The most common version of the K-Means technique is known as Lloyd-Forgy algorithm which begins by Forgy initialization [36],[37]. The initial codewords are selected randomly from the original image blocks by Lloyd-Forgy method. The technique aims to find an optimum codebook that consists of codeword vectors list representing the image data best. The K-Means technique starts its process by dividing the original input image into sub-blocks. Consider the original image $Y=\{X_{ij}\}$ comprises $N\times N$ pixels and is divided into sub-blocks of size $m\times m$ pixels. The number of sub-blocks is determined by $N_b = \left(\frac{N}{m}\times\frac{N}{m}\right)$. All image data is represented by a set of image vectors $X = \{x_i, i = 1,2,\ldots, N_b\}$. Let the variable L represents the size of the blocks, where $L=m\times m$ pixels. Therefore, each sub-block x_i is defined within an L-dimensional Euclidean space, denoted as $x_i\in\Re_L$. Then, a codebook of N_c codewords, $C=\{c_1,c_2,\ldots,c_{N_c}\}$, $c_j\in\Re_L$, $j=1,2,..,N_c$ represents the original image. Each image vector is denoted as a row vector by $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iL})$ and the jth codeword in C is denoted by $c_j = (c_{j1}, c_{j2}, c_{j3}, \ldots, c_{jL})$. The K-Means algorithm maps each original image block to a codeword by using minimum square error (MSE) criteria through a number of iterations. Once the algorithm finishes, the index number of the related codeword is used instead of the image block itself to represent the data in a block. Thus, a VQ-based image compression is achieved by forming the vectors of C that minimize the MSE value given in Equations (1)-(3).

$$\mathrm{MSE(C)} = \frac{1}{N_b}\sum_{j=1}^{N_c}\sum_{i=1}^{N_b}\mu_{ij}\left\|x_i - c_j\right\|^2 \tag{1}$$

$$\sum_{J=1}^{N_c}\mu_{ij} = 1\,, \qquad i\in\{1,2,\ldots,N_b\} \tag{2}$$

$$\mu_{ij} = \begin{cases} 1\,, & \text{if } x_i \text{ is in the } j^{\text{th}} \text{ cluster} \\ 0\,, & \text{otherwise} \end{cases} \tag{3}$$

The Euclidean distance D between the ith image block and jth codeword is defined by $\left\|x_i - c_j\right\|$.

The two rules specified by Equation (4) and Equation (5) are applied by the K-Means algorithm. The group of image vectors (R_j, j=1,2, … N_c) must satisfy the distance condition

$$R_j\supset\left\{x\in X\colon D\left(x,c_j\right) < D(x,c_k), \forall k\neq j\right\}. \tag{4}$$

The center of R_j denoted by c_j is calculated using

$$c_j = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i, \qquad x_i\in R_j \tag{5}$$

where N_j is the number of elements in R_j. Let the image vectors be x_i, i = 1,2, …, N_b, and the initial codewords determined by the Forgy method are $c_j(0)$, j = 1, 2,…,N_c. Then, the K-Means algorithm applies the four steps given below to determine a suboptimal codebook that is used by an optimization algorithm.

Step 1: Select the number of N_c cluster centers from the original image blocks randomly. This process is named as Forgy method,

Step 2: Assign the original image blocks to the corresponding cluster center μ_{ij} using Euclidean distance criteria. The centers of clusters are saved in an indicator matrix with size of $N_b\times N_c$ pixels.

$$\mu_{ij} = \begin{cases} 1, & \text{if } D\left(x_i, c_j(k)\right) = \min D\left(x_i, c_j(k)\right) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Step 3: Calculate the new cluster centers given by Equation (4).

Step 4: Execute the Equations (4) and (6) sequentially until the cluster centers of c_j do not change.

The codebook structure and image reconstruction are shown in Figure 1.

2.2 PSO algorithm

The Particle Swarm Optimization (PSO) is a population-based metaheuristic algorithm inspired by the social behavior of animals like birds flocking, fish schooling, and insect swarming. The PSO was introduced by Kennedy and Eberhart in 1995 [38]. It is simpler to implement than other metaheuristic and evolutionary algorithms and requires only a few parameters to be tuned. The PSO was modified by Shi and Eberhart [39] and applied to a broad spectrum of optimization problems [40].

All optimization problems require evaluations of their own objective functions by the solver algorithms to reach the global

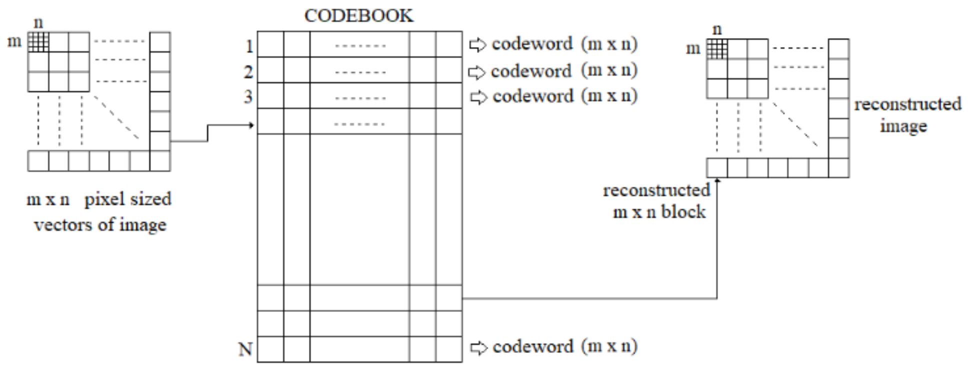

Figure 1. Original image blocks, codebook structure and reconstructed image structure of VQ [17].

optimum solution of the unknown parameters in an iterative manner. An individual of the PSO algorithm is represented by a particle resembling for example, a migratory bird who has its own position and velocity in the swarm. The position of each particle is viewed as a potential solution to the optimization problem while its velocity indicates the distance it will move in the next iteration. Consequently, the velocity and position of each particle are updated after each iteration to adjust the parameters and achieve the global best position. The PSO algorithm inherently achieves both global and local searches. The best position of each particle in the swarm is denoted by PB^i $(i = 1,2, \dots ,p)$ where *i* indicates the particle number and p is the size of the swarm. The best position of the population (that is of *gbest*) is denoted by GB that is recorded along the iterations. At the beginning, the position and velocity of each particle in the swarm are initialized to random values from the search space of the problem solution. In an iteration t, the PSO algorithm determines the direction of each particle in the swarm using the velocity and position equations expressed by Equations 7 and 8, respectively, where $t + 1$ points to the next iteration.

$$V_{t+1}^i = k_w V_t^i + k_p r_1 (PB_t^i - X_t^i) + k_g r_2 (GB_t - X_t^i) \quad (7)$$

$$X_{t+1}^i = X_t^i + V_{t+1}^i \quad (8)$$

Here, X and V represent the position and the velocity of the particle, respectively, in the multidimensional search space.

Table 1. Pseudo-code of the basic PSO algorithm.

```
Initialize position $X_1^i$ and velocity $V_1^i$ of all particles
Define $PB_0^i = X_1^i,\ GB_1 = X_1^1$;
for $t = 1$ to $T$ do
    for Each particle in the swarm do
        Evaluate objective function: $E = f(X_t^i)$;
        if $E$ is better than $f(PB_{t-1}^i)$ then
            $PB_t^i \leftarrow X_t^i$;
        end if
        if $E$ is better than $f(GB_t)$ then
            $GB_t \leftarrow X_t^i$;
        end if
    end
    for Each particle in the swarm do
        $V_{t+1}^i = k_w V_t^i + k_p r_1 (PB_t^i - X_t^i) + k_g r_2 (GB_t - X_t^i)$;
        if $|V_{t+1}^i| > V_{max}$ then clamp it to $V_{max}$; end if
        $X_{t+1}^i = X_t^i + V_{t+1}^i$;
    end
end
```

The last two terms in Equation 7 are called cognitive and social components, respectively, that contribute to the velocity and position updates of each particle for the next iteration. The constant k_w is a user defined value called inertia weight [38]. The others, k_p and k_g are acceleration constants which determine the degree of influence from the particle's own best experience and current global best particle in the swarm. Finally, r_1 and r_2 are two independent random numbers having uniform distribution in the interval of [0,1]. The pseudo code of the standard PSO algorithm is given in Table 1 where T is the maximum number of iterations chosen as the termination criterion and $f(X^i)$ is the evaluation result of the objective function for the position of particle *i*. The optimal solution is equal to GB when the algorithm stops. Recall that the objective function in our problem is MSE between the reconstructed image obtained by the codebook list $\{X^i; i = 1,2, \dots ,p\}$ and the original image. The pseudo-code of the PSO algorithm is given in Table 1.

2.3 Simultaneous perturbation stochastic approximation algorithm

Gradient-based optimization algorithms suppose that information exists about the gradient of the objective function to optimize the parameters. But in many real-world optimization problems, gradient cannot be computed or observed directly. This difficulty led to the development of stochastic approximation (SA) algorithms. They rely on an approximation to the gradient that is achieved by only measurements of the objective function in a stochastic setting. Convergence properties of many such gradient-free stochastic algorithms are similar to those of the gradient-based algorithms [41]. SPSA is a stochastic optimization algorithm produced by Spall [20]. The SPSA algorithm has a superior feature to the other optimization methods. This feature is estimating the gradient function value when the exact gradient is not available for the objective function. Therefore, the SPSA technique increases the quality of the unknown parameters by estimating the gradient of the objective function and using it to update the unknown parameters.

All deterministic multivariate optimization algorithms aim at minimizing a differentiable objective function $f(\theta)$ by searching for zero-gradient point. When gradient g of the function cannot be directly computed in implementations (e.g., in many real-world problems), the SA algorithms can be used alternatively to update the solution vector θ_t in d-dimensional search space by

$$\theta_{t+1} = \theta_t - a_t \hat{g}_t(\theta_t) \quad (9)$$

where, $\hat{g}_t$ is an approximation to the gradient vector g at iteration t and a_t is a scalar gain sequence that has to meet certain conditions [20]. The approximation to the gradient vector is obtained using simultaneous random perturbations and two measurements of the objective function by

$$\hat{g}_t(\theta_t) = \frac{f(\theta_t + c_t\Delta_t) - f(\theta_t - c_t\Delta_t)}{2c_t}\begin{bmatrix}\Delta_{t,1}^{-1}\\ \Delta_{t,2}^{-1}\\ \cdot\\ \cdot\\ \cdot\\ \Delta_{t,d}^{-1}\end{bmatrix} \qquad (10)$$

where each element $\Delta_{t,i}$ of the vector Δ_t, takes on a value of +1 or -1, as generated by a zero-mean Bernoulli distribution, and c_t is a positive gain sequence. Both gain sequences are computed as follows.

$$a_t = a/(A+t)^{\alpha} \qquad (11)$$

$$c_t = c/t^{\gamma} \qquad (12)$$

These equations introduce five SPSA constants. In [41], Spall recommends using the following values for the three of them: the stability constant $A = 60$, $\alpha = 0.602$, and $\gamma = 0.101$. However, he deduces that the performance of the SPSA is very sensitive to the choice of both gain sequences. He adds that this problem occurs in other stochastic optimization algorithms as well, due to their own coefficients. Therefore, it is left to researchers to find suitable values for the constants a and c according to the optimization problem to be tackled. The pseudo-code of the SPSA algorithm is given in Table 2.

3 The proposed algorithm and its implementation on codebook optimization

3.1 The proposed algorithm: SPSA-FPSO

The algorithm proposed in this paper has an embedded structure where the SPSA algorithm performs a guidance task for the particle *gbest* only in the swarm of PSO. Thus, the combined process should be considered as a single algorithm in its own. The proposed algorithm is inspired from [35] where the SPSA process creates an alternative global best (*agbest*) particle from the positon of the particle *gbest*. If the position of *agbest* gives a better objective function value than that of the *gbest*, then it is accepted that SPSA provided a new position towards the global optimum. Next step is to replace the position of *gbest* with the position of *agbest*, thus moving it one step further towards the global optimum or perhaps moving it away from a local optimum where it is likely to get stuck. Of course, no position replacement occurs if *agbest* has a worse position. SPSA is given the opportunity to provide this guidance at each iteration, which can eventually lead the swarm to the global optimum or a point very close to it. The major difference of our algorithm from the one summarized above lies in the number of particles created by SPSA. In this paper, multiple particles instead of one, each of which is a candidate for being *agbest* are put forward as novelty. In the proposed algorithm, SPSA is applied over the position *GB* for *s* times, independently, creating an ensemble of particles with a variety of positions offering more diversity starting from this position. The particle *agbest* is determined easily after the objective function evaluations for all alternative particles in the ensemble. Let us denote the position of *agbest* by *AGB*. Now, *AGB* being an elite position will compete with *GB* of *gbest,* and whichever is the better position will be eligible to become the new *gbest*. That is, if *AGB* scores a better function evaluation value, then it will replace *GB*. Note that this strategy using SPSA only deals with *gbest* and creates some competitive particles outside the swarm. Except for this difference, the internal PSO process is kept the same. Table 3 gives the pseudo code of the proposed algorithm. You can see

Table 2. Pseudo-code of the SPSA algorithm.

```
Initialize solution vector $\theta_1$;
Set $A = 60$, $\alpha = 0.601$, and $\gamma = 0.101$;
Set $a$ and $c$ to values suitable to optimization problem;
for $t = 1$ to $T$ do
    • Generate $d$-dimensional Bernoulli distributed
      perturbation vector: $\Delta_t$
    • Let $a_t = a/(A+t)^{\alpha}$ and $c_t = c/t^{\gamma}$
    • Compute $f(\theta_t + c_t\Delta_t)$ and $f(\theta_t - c_t\Delta_t)$
    • Compute $\hat{g}_t(\theta_t)$ using Equation (10)
    • Compute $\theta_{t+1}$ using Equation (9)
end
```

Table 3. Pseudo-code of the SPSA-FPSO algorithm.

```
Initialize position $X_1^i$ and velocity $V_1^i$ of all particles
Define $PB_0^i = X_1^i$, $GB_1 = X_1^1$; Set size $s$;
Set $A = 60$, $\alpha = 0.601$, and $\gamma = 0.101$;
Set $a$ and $c$ to values suitable to optimization problem;
for $t = 1$ to $T$ do
    for Each particle in the swarm do
        Evaluate objective function: $E = f(X_t^i)$;
        if $E$ is better than $f(PB_{t-1}^i)$ then
            $PB_t^i \leftarrow X_t^i$;
        end if
        if $E$ is better than $f(GB_t)$ then
            $GB_t \leftarrow X_t^i$;
        end if
    end
    Let $a_t = a/(A+t)^{\alpha}$ and $c_t = c/t^{\gamma}$;
    Define $\theta = GB_t$;
    Set $F$ to an extreme value with respect to optimum of
    the objective function;
    for $k = 1$ to $s$ do
      • Generate $d$-dimensional Bernoulli distributed
        perturbation vector: $\Delta_t$
      • Compute $f(\theta + c_t\Delta_t)$ and $f(\theta - c_t\Delta_t)$
      • Compute $\hat{g}_t(\theta)$ using Equation (10)
      • Compute position of $k^{\text{th}}$particle: $Y^k = \theta - a_t\hat{g}_t(\theta)$
        if $f(Y^k)$ is better than $F$ then
            $AGB \leftarrow Y^k$ and $F = f(Y^k)$;
        end if
    end
    if $f(AGB)$ is better than $f(GB_t)$ then
        $GB_t \leftarrow AGB$;
    end if
    for Each particle in the swarm do
        $V_{t+1}^i = k_w V_t^i + k_p r_1(PB_t^i - X_t^i) + k_g r_2(GB_t - X_t^i)$;
        if $|V_{t+1}^i| > V_{max}$ then clamp it to $V_{max}$;
        end if
        $X_{t+1}^i = X_t^i + V_{t+1}^i$;
    end
end
```

that the basic steps of both algorithms are included in each iteration. The innermost loop in the algorithm performs creating an ensemble of alternative particles by SPSA and determining *agbest* in the ensemble. At the beginning of the SPSA steps of the algorithm, the gain sequences a_t and c_t are updated accordingly before a new ensemble of alternative particles is generated.

The idea of using multiple SPSA-generated positions, instead of one, yields more increased diversity in the search space, proportional to the size *s*, thus granting better ability to escape from local optimum. Recall that SPSA can approach the optimum value faster than other stochastic optimization methods by simultaneously perturbing all parameters based on the approximate gradient of the objective function. The PSO and similar metaheuristic algorithms using only the exploration and exploitation ability of the population require a high computational load per iteration since they have to evaluate the objective function as many times as the number of individuals in the population. Therefore, improving metaheuristic algorithms to reach the optimum value with fewer iterations is of great importance in practice, especially when the number of unknowns is very large. Our algorithm uses the fast gradient approximation (owing to simultaneous perturbations) of the SPSA in the calculation of a better next position (codebook) for *gbest* by making different gradient approximations *s* times and selecting the best one. The classical PSO which does not use gradient information can benefit from this guidance provided by a reliable gradient approximation. It is obvious that, by SPSA-FPSO, the optimum solution can be reached with many fewer iterations due to such a valuable guidance to the *gbest*, which is known to be the only guide by the swarm.

3.2 Implementation on codebook optimization

Figure 2 illustrates the block-based lossy image compression system including the proposed optimization algorithm used for codebook estimation. At the beginning, the original image data is converted to a list of vectors based on 4×4 image blocks as explained in Section 2.1. Then, a separate initial codebook is prepared for each particle using the standard K-Means algorithm while velocity of each particle is initialized randomly. After the initialization, the iterative algorithm given in Table 3 is applied to find the optimal codebook. In each iteration, the codebook *gbest* evolved by the PSO rules is passed to the SPSA algorithm where it produces an ensemble of size *s* new codebooks each of which corresponds to an alternative particle. Among the SPSA generated ensemble, the best one is selected and recorded as *agbest*. If the codebook of *agbest* is better than that of *gbest*, then it replaces the poorer codebook. Finally, the PSO swarm of codebooks are updated by Equations (7) and (8) for the next iteration. When the iteration number reaches to the maximum number, the best codebook of the swarm (*gbest*) is determined by the SPSA-FPSO. To achieve a lossy compression, each image block is simply replaced by index of the relevant codeword in the best codebook. This process results in an indexed image which is further encoded for a communication channel by a lossless technique such as arithmetic coding or Huffman coding. After the encoding process, the best codebook and the encoded image are sent over the channel. The indexed image is extracted after channel decoding at the receiver end. Finally, the reconstructed image is obtained by using the best codebook.

4 Experimental work

The experimental study presented in this section was conducted to obtain and compare the performance of codebook estimation of the base algorithms and the proposed hybrid algorithm on a test image set. In addition, the effect of *s* on the performance of the hybrid algorithm was also investigated.

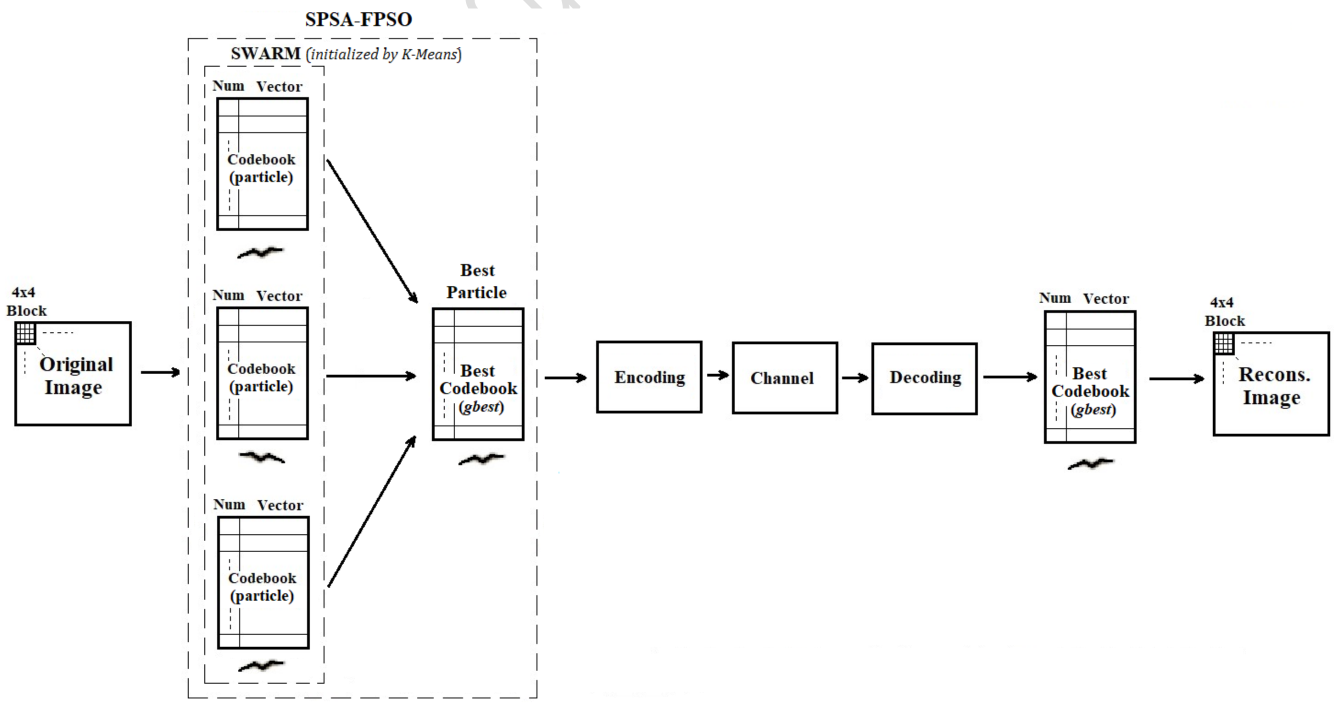

Figure 2. Block diagram of the VQ-based lossy image compression system using SPSA-FPSO.

4.1 Test images and experimental arrangements

In the experiments, six gray level test images, known from the image processing research field were used. The test images are of 256×256 pixels size and show different contrast levels and degree of detail. Of these, *Aerial* and *Barbara* are categorized as high contrast images. While *Cameraman*, *Lena* and *Peppers* have a fair contrast, the *Clock* image presents a low contrast with uneven distribution of details. To allow visual comparison of test results, only two representative samples were selected for the sake of space saving: *Lena* with medium contrast and *Clock* with low contrast and uneven detail distribution. The results are shown in the Section 4.2.

The experiments were conducted as follows. Each original test image divided into 4×4 pixel image blocks were converted to a list of vectors of size 16×1 with 256 gray levels. Thus, each image consists of 4096 vectors to be quantized and encoded, in total. A unique initial codebook for each particle in the swarm was computed by the K-Means algorithm based on the Forgy selection method [36]. The initial codebooks were also recorded for using in every run of the algorithms on each test image. For the PSO and SPSA-FPSO algorithms, all particles' initial positions and the initial best positions were set to these initial codebooks. Similarly, the solution vector θ_1 in the SPSA algorithm was also initialized to the first recorded codebook.

Since the visual error of the reconstructed image is more evident at high compression ratios, a codebook size of 8 codewords was preferred in all experiments. This codebook size corresponds to the image compression ratio of 0.203 bpp. Therefore, the number of unknowns (i.e., pixel gray level values) in any codebook was 128 since each codeword has 16 unknown gray levels. Thus, an optimum codebook for any test image must be found by any algorithm in the 128-dimensional search space. The MSE given by Equations (1)-(3), between an original image and its reconstructed form obtained from any individual codebook C were computed in the optimization algorithms. Each algorithm tested in the experiments tried to converge to the optimum codebook with a global minimum MSE value for each test image. However, when the maximum number of iterations was reached, the goodness of their estimation was revealed by the difference between the MSE of the final codebook they could produce and the global minimum. For all test runs on all the algorithms, the maximum number of iterations was determined as T=1500. This number was used as the single termination rule for every run of an optimization algorithm as seen in Tables 1-3.

The parameters of the optimization algorithms were chosen as follows. The number of particles (i.e., size p) in both the PSO and SPSA-FPSO algorithms was set to 100. The number of alternative particles in the SPSA-FPSO algorithm was set to different values of s=1, 3, 10, 25, and 50, for performance comparisons. The user-defined parameter of k_w was selected as 0.7 while k_p and k_g were determined as 1.0 and 2.0, respectively. The velocity of the particles was restricted to $V_{max} =$ 10 for both the PSO and SPSA-FPSO algorithms. For A, α and γ in the SPSA algorithm, the recommended values were used as 60, 0.602 and 0.101, respectively, while a and c were set to 15 and 0.25, respectively. Because the last two parameters must be determined with respect to the optimization problem and the range of values for the unknowns, they were adjusted to best values after conducting a series of trial runs. However, for the SPSA-FPSO algorithm, we needed to dynamically change the value of a during the iterations to boost the acceleration ability of the proposed hybrid algorithm. While some smaller value of a contributes to the acceleration of convergence at early iterations, a larger value as in the SPSA algorithm strikingly reduces the probability of getting stuck in a local minimum in the subsequent iterations. A function of t for a was determined empirically to use in the SPSA-FPSO algorithm as given in Equation (13).

$$a = t/50 + 5 \tag{13}$$

The equation should be inserted into the pseudo-code in Table 3 before the line where the gain sequences a_t and c_t are computed.

4.2 Test results

Here, the proposed SPSA-FPSO algorithm is compared to the basic PSO and SPSA algorithms in an experimental framework. The visual results of the base algorithms and the proposed algorithm for the two selected test images are shown in Figures 3 and 4 after a test run for each algorithm was completed. For *Lena*, when the last iteration was completed, the SPSA and PSO algorithms reached the unsatisfactory values of MSE=368.7 and MSE=329.6, respectively while the proposed SPSA-FPSO algorithm achieved the almost global optimum value of MSE=323.5 with only three alternative particles (i.e., s=3). When the low contrast image *Clock* was used by the algorithms, the SPSA-FPSO algorithm reached the value of MSE=271.9 which is nearly global minimum again. On the other hand, SPSA and PSO appeared with MSE=305.9 and MSE=281.9, respectively at the last iteration, still struggling for the global optimum. This superiority of the proposed algorithm over the others can also be visually noticed in the reconstructed images shown in Figures 3 and 4.

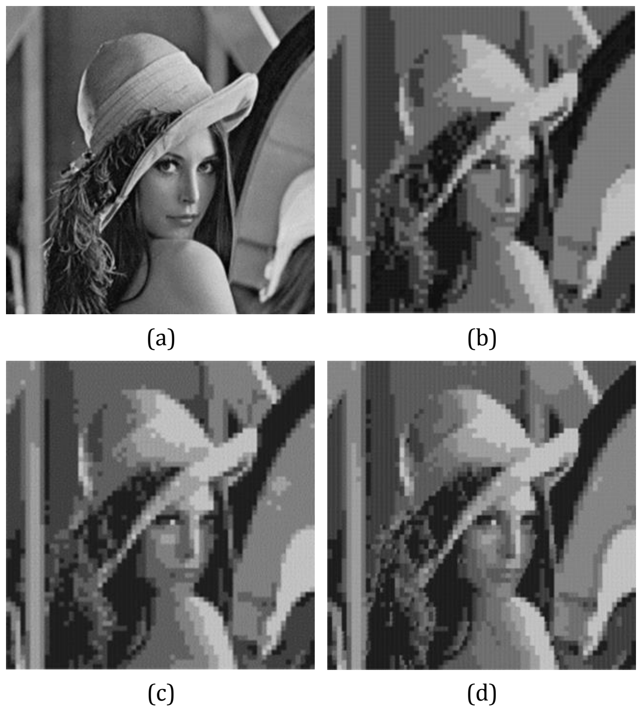

Figure 3. Visual comparison of the reconstructed images for *Lena*. (a) Original, (b) SPSA result (MSE=368.7), (c) PSO result (MSE=329.6), and (d) SPSA-FPSO result for s=3 (MSE=322.0).

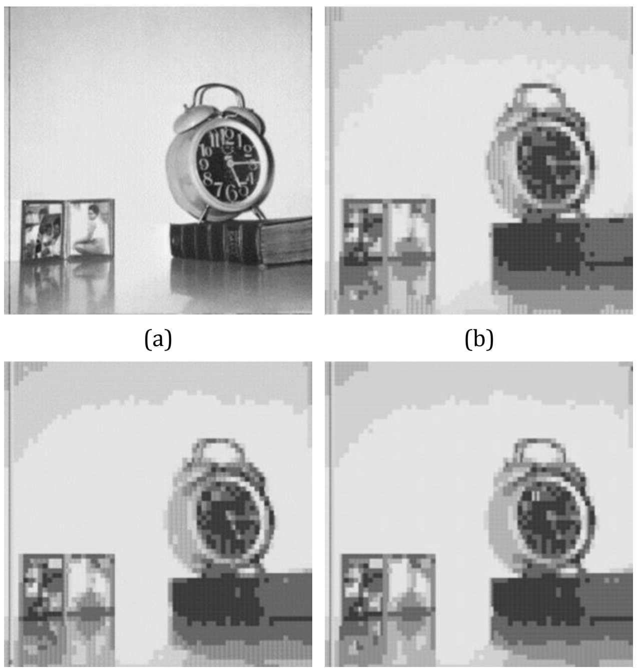

Figure 4. Visual comparison of the reconstructed images for *Clock*. (a) Original, (b) SPSA result (MSE=302.3), (c) PSO result (MSE=281.7), and (d) SPSA-FPSO result for *s*=3 (MSE=271.9).

The convergence curves of the all algorithms used to estimate the optimum codebooks for the all test images are shown in Figure 5. Here, each algorithm was run nine times for each of the test images. Then, the median value of the nine MSE curves was computed point by point to obtain an average curve. Therefore, every curve plotted in Figure 5 is actually a median average and represents the convergence characteristics of an algorithm with a fair approximation. The PSO algorithm either got stuck at a local minimum or fails to complete its convergence when the maximum number of iterations was reached for all images except *Aerial* and *Peppers*. On the other hand, the SPSA-FPSO algorithm reached the global minimum or settled at a very close value for all images except *Cameraman*. For this image, the proposed algorithm converged to a nearby value only with *s*=50. This is a difficult image for optimization algorithms used in codebook estimation of block-based compression. The image contains long strong edges at many different orientations. This forces the optimization algorithms to concentrate on those edges, leading to the ignorance of the rest of the image when MSE is the objective function. As a result, the algorithms mostly got trapped into a local minimum for this image. Note that the proposed algorithm converges quickly to the global minimum even for small values of *s* (i.e., 3 and 10). These results prove that the SPSA guidance provided by several alternative particles increases the diversity of the basic PSO in the search space and moves *gbest* to a more efficient position in each iteration. In short, we can say that SPSA can guide (or drive) *gbest* better than PSO can.

The descent rate in the convergence curves of the SPSA-FPSO algorithm, especially in the early iterations is also remarkable. Considering only the number of iterations, the proposed algorithm clearly outperforms the base algorithms in terms of convergence speed. The fact that SPSA lags far behind other algorithms in the descent of convergence curves can be explained by its low exploration effort per iteration (one new codebook generation only). This algorithm needs more iterations for convergence, although this does not mean that it needs more time. On the contrary, it may require less time when the number of function evaluations is taken into account. However, it is shown in [35] that the SPSA algorithm fails if the target function has too many local optima or the number of unknown parameters is large.

An overview of the results regarding the convergence accuracy and speed on the test images is given in Table 4. The optimal MSE values (MSE$_{op}$) obtained with $N_C = 8$ for each image are appended next to the image name and shown underlined. For each algorithm, the median of the final MSE values is written in gray cells. It is seen that SPSA-FPSO can rapidly converge to the corresponding MSE$_{op}$ for all images except *Cameraman*. However, note that all the algorithms fail to reach the MSE$_{op}$ value of this image due to its characteristics explained above. To simplify the comparison of the convergence accuracies, a near optimal MSE value was chosen at 1% above the global optimum and represented by MSE$_{no}$. All the median MSE values that are less than or equal to MSE$_{no}$ (i.e., within 1% tolerance) were considered accurate and shown in bold. In conclusion, it can be said that SPSA-FPSO is clearly superior to the base algorithms when the convergence accuracy is taken into account. It should also be noticed that the effect of *s* on the convergence accuracy of the SPSA-FPSO algorithm is small. However, the opposite is true when the convergence speed is concerned. To compare the convergence speeds, the numbers

Table 4. Results of convergence accuracy (final MSE) and iteration number (t_{no}) on test images.

Image name	$\underline{\text{MSE}_{op}}$ / MSE$_{no}$	SPSA	PSO	SPSA-FPSO *s*=1	*s*=3	*s*=10	*s*=25	*s*=50
Aerial	$\underline{518.4}$	529.1	**523.4**	**522.6**	**520.5**	**519.6**	**519.6**	**519.8**
	523.6	-	975	268	149	140	97	89
Clock	$\underline{271.4}$	303.3	280.5	**272.5**	**271.9**	**274.0**	**271.9**	**272.2**
	274.1	-	-	412	362	399	218	133
Barbara	$\underline{338.6}$	**339.8**	**342.4**	**338.9**	**338.9**	**338.8**	**338.7**	**338.8**
	342.0	934	1500	142	117	67	36	19
Peppers	$\underline{309.5}$	317.2	**310.1**	**309.6**	**309.6**	**309.6**	**309.6**	**309.8**
	312.6	-	659	375	224	114	71	44
Lena	$\underline{322.8}$	348.8	330.4	**323.3**	**323.4**	**323.6**	**323.6**	**323.6**
	326.0	-	-	623	512	253	150	99
Cameraman	$\underline{405.5}$	447.8	423.1	422.7	422.3	423.0	425.2	414.8
	409.6	-	-	-	-	-	-	-

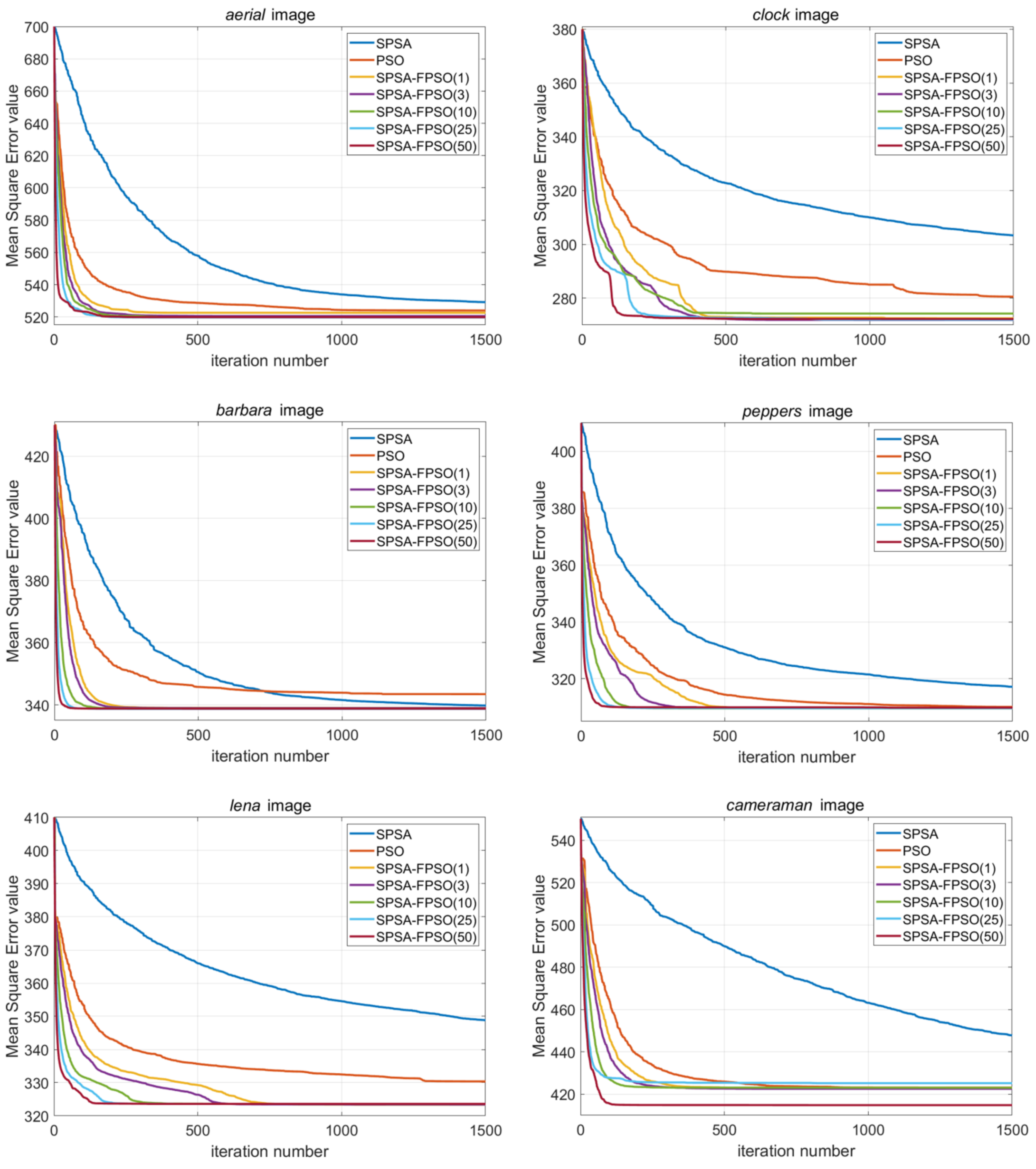

Figure 5. Convergence curves of SPSA-FPSO with different s values (given between parenthesis) on codebook estimation for six test images ($N_C = 8$).

in the white cells in Table 4 can be used. They are the iteration numbers, denoted by t_{no}, at which the algorithms reached the MSE$_{no}$ value for each image. Each t_{no} number is the median of such iteration numbers found over nine runs. Let us take the *aerial* image for example. The table points out that PSO reach MSE$_{no}$ at 975th iteration while SPSA cannot achieve it in 1500 iterations. The proposed algorithm reaches the same value at 268th, 149th, 140th, 97th and 89th iterations for the values of s =1, 3, 10, 25 and 50, respectively. When all the t_{no} numbers are examined, it is seen that the SPSA-FPSO algorithm converges much faster than both PSO and SPSA algorithms for all test images. Even with s =1, the proposed algorithm can reduce the number of iterations required for the convergence of the PSO algorithm quite a lot. In addition, the experimental results show

Table 5. Computational complexity comparison: PSO vs. SPSA-FPSO on the test images.

Image name	PSO n_f	Computational advantage of SPSA-FPSO over PSO				
		s=1	s=3	s=10	s=25	s=50
Aerial	97500	3.53	6.00	5.36	5.74	4.38
Barbara	150000	10.26	11.76	17.22	23.81	31.58
Peppers	65900	1.71	2.70	4.45	5.30	5.99

Table 6. Computational complexity comparison: single vs. multiple alternative particles in SPSA-FPSO.

Image name	SPSA-FPSO (s=1) n_f	Computational advantage of SPSA-FPSO with multiple alternative particles			
		s=3	s=10	s=25	s=50
Aerial	27604	1.70	1.52	1.63	1.24
Clock	42436	1.08	0.82	1.11	1.28
Barbara	14626	1.15	1.68	2.32	3.08
Peppers	38625	1.58	2.61	3.11	3.51
Lena	64169	1.15	1.95	2.44	2.59

that as the number of alternative particles (codebooks) increases, the speed in iteration axis also increases. This result clearly indicates the effectiveness of multiple alternative particles generated by SPSA in our hybrid algorithm for increasing diversity and providing accurate guidance in a search space.

4.3 Comparison of computational loads

When computational complexities of optimization algorithms are compared, considering the number of objective function evaluations instead of the iteration number at the global optimum is a more accurate approach. For most algorithms, the computational cost of a single objective function evaluation is much heavier than the implementation cost of the optimization algorithm itself. This is also true for the proposed algorithm and the base algorithms since MSE computation between an original image and a reconstructed image by using a codebook is relatively long. Assuming that the necessary function evaluations are stored in memory once computed, the PSO algorithm requires p function evaluations while SPSA-FPSO requires $p + 3s$ evaluations per iteration. Although this relationship makes SPSA-FPSO seem to be computationally expensive, the reality is different because SPSA-FPSO converges earlier by the iteration number. Therefore, it is more sensible to find the total number of function evaluations that each algorithm does until it reaches the iteration number t_{no}. Then, the costs of the algorithms can be compared according to their number of function evaluations. The total number of function evaluations, denoted by n_f, for PSO and SPSA-FPSO until they reach the associated iteration number t_{no} given in Table 4 is computed by

$$n_f = (p + 3s)t_{no} \tag{14}$$

where n_f for PSO is found by taking $s = 0$. Let us define experimental computational advantage as the ratio of the n_f value for PSO to the n_f value for SPSA-FPSO. The computational advantage of SPSA-FPSO over PSO for varying number of alternative particles are listed in Table 5 where the n_f value for PSO on each test image is given next to the image name. Comparisons were made only for test images where PSO reached MSE$_{no}$. One can notice that the computational advantage is unsuprisingly image dependent and can be up to order of 30 times.

To investigate the effect of using multiple alternative particles on the convergence speed in the time dimension, the computational advantages of SPSA-FPSO with multiple particles over single particle use were obtained and listed in Table 6. Here, the n_f value for $s = 1$ on each test image is given next to the image name and taken as the reference in the calculations of the computational advantages. As seen from the table, the number of multiple alternative particles increases the computational advantage of SPSA-FPSO (i.e., less run time) compared to using a single particle. However, the increase in the computational advantage by the number of alternative particles can be high or remain within marginal limits depending on the image content.

4.4 Performance comparison to other metaheuristic algorithms

Finally, some tests were conducted to compare the performance of the SPSA-FPSO algorithm with the performances of other well-known metaheuristic algorithms in the application of codebook optimization. Bat algorithm (BA), firefly algorithm (FA), genetic algorithm (GA) and improved adolescent identity search algorithm (IAISA) were included in these tests, respectively. The parameters of the SPSA-FPSO algorithm were kept the same as given in Section 4.1. The tests were performed by selecting the number of alternative global particles as 3 and 25. In all algorithms, the population size and codebook size were fixed as 100 and 8, respectively. The parameter values of other metaheuristic algorithms were selected as given in Table 7, which are considered appropriate in [17] for the codebook optimization problem. Convergence curves of all algorithms were obtained for the two selected test images. The results are given in Figure 6. It is seen that SPSA-FPSO is superior not only to PSO algorithm but also to other metaheuristic algorithms in terms of both convergence accuracy and convergence speed.

Table 7. Parameters of the metaheuristic algorithms

Algorithm	Parameter values
FA	α=0.5, γ=0.01, $\beta_0 = 0.9$
BA	α=0.997, γ=0.996, [min max] frequencies: [0,1] initial values (random): frequency [0,1], pulse rate [0.5, 1], loudness [0,1]
GA	crossover rate=10%, mutation rate=3%, selection method is K-Tournament (K=3)
IASIA	search diameter=0.025, Chebyshev polynomial degree=3

5 Conclusions

The use of VQ offers an effective approach to lossy image compression by mapping image blocks into codebook vectors, significantly reducing the image size. Key to the success of VQ-based compression is the estimation of an optimal codebook, which directly impacts the quality of the reconstructed image. Optimal codebook generation through metaheuristic algorithms has gained significant attention in many research areas. Although the hybridization of these methods, particularly in a coupled way to enhance both convergence speed and accuracy has been applied in various research areas, it has not received much attention in solving the codebook estimation problem.

The motivation for this work is to overcome the inherit diversity limitations of PSO by hybridizing it with SPSA which can generate new codebooks by fast gradient approximations to the objective function MSE. In the proposed hybrid algorithm, the codebook owned by *gbest* of PSO is passed to the SPSA algorithm to have it generate an ensemble of *s* new codebooks, so-called alternative global codebooks, to guide *gbest* further. The best of the alternative codebooks in the ensemble replaces, (if it is better than) that of *gbest* to provide a more efficient guidance to the entire swarm in the next iteration. The second contribution of this work is the presentation and discussion of the results obtained from the first-time application of this high performance optimization algorithm to codebook estimation in VQ-based lossy image compression. The results of the experiments conducted on six test images highlight the superiority of the SPSA-FPSO algorithm over the base algorithms PSO and SPSA in the context of VQ-based image compression. By embedding the SPSA within the PSO framework, the hybrid algorithm successfully reduces the premature convergence issues typically observed in PSO, enabling it to escape local minima and reach or approximate the global minimum. It is shown that the SPSA-FPSO algorithm increases the diversity by using multiple alternative codebooks and provides more accurate guidance in the search space. The algorithm not only achieves lower MSE values for the test images, but also significantly improves the convergence speed without sacrificing accuracy. The proposed algorithm reaches or settles very close to the global minimum for all test images except a troublesome image. However, the best performer among the algorithms in competition happens to be SPSA-FPSO again. Especially in the early iterations, the speed at which the proposed algorithm approaches the global minimum is remarkable, which can also make it a good global optimization partner for a faster local optimizer.

When the computational loads are compared, it is seen that SPSA-FPSO algorithm requires a smaller number of objective function evaluations than PSO does and therefore the total computational cost is lower. However, the computational advantage offered by the algorithm is data dependent and cannot be foreseen exactly. Although the computational advantages experimented from the test images may give a rough idea, there are also other issues affecting the range of figures, namely, selection of the initial codebook and optimization of the algorithm parameters. The experimental results also show that the number of alternative global codebooks (i.e., value of *s*) positively influence the convergence speed while it has a small or trivial effect on accuracy. However, the computational advantage gained by increasing the number of alternative particles can either be significant or minimal, depending on the image content.

The study concludes that the enhanced convergence features of SPSA-FPSO make it a promising method for optimizing codebook in VQ-based lossy image compression tasks. It outperforms not only the basic PSO but also other well-known metaheuristic algorithms. Considering that the target of this

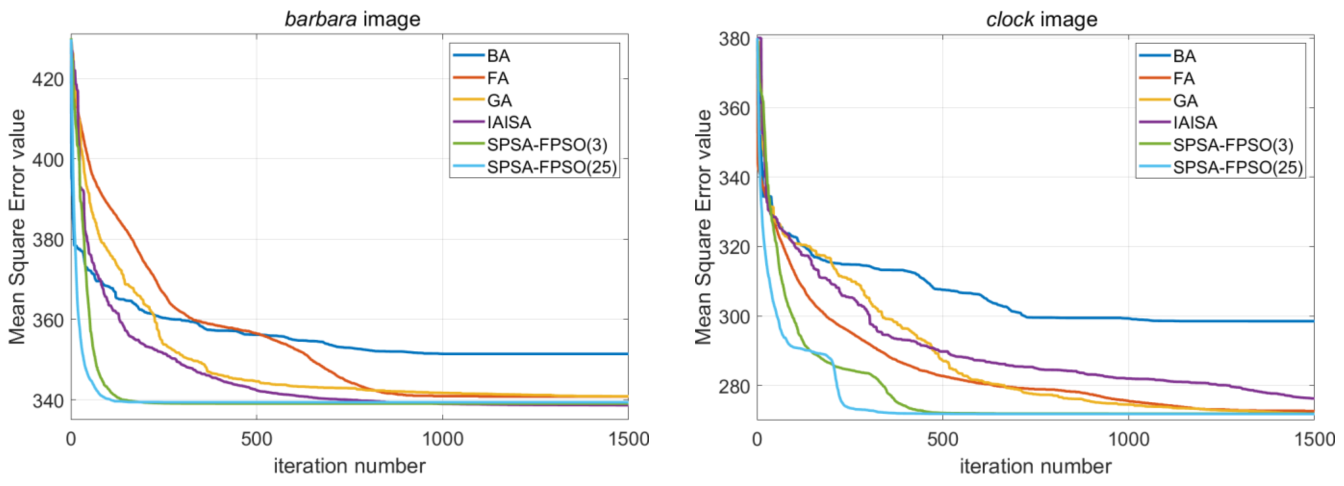

Figure 6. Convergence curves of the metaheuristic algorithms for comparison to SPSA-FPSO on codebook estimation for two test images ($N_C = 8$).

research is image compression, the results could potentially pave the way for the utilization of the suggested approach in related areas that prioritize compression and optimization, including data compression and signal processing. From a broader perspective, the findings of this work suggest that the SPSA-driven PSO algorithm employing an ensemble of alternative global particles can also be effectively applied to optimization problems in other research areas, offering a fast and stable solution especially for complex, high-dimensional problems.

6 Author contribution statements

In the scope of this study, the first author contributed by reviewing the literature, forming the methodology, developing the software, performing formal analysis, and writing the manuscript. The second author contributed by reviewing the literature, forming the idea and methodology, supervision of the experiments, performing data curation, and writing the manuscript.

7 Ethics committee approval and conflict of interest statement

There is no need to obtain permission from the ethics committee for the article prepared. There is no conflict of interest with any person / institution in the article prepared.

8 References

[1] Gray RM. "Vector Quantization". *IEEE ASSP Magazine*, 1(1), 4-29, 1984.

[2] Wu Z, Yu J. "Vector quantization: a review". *Frontiers of Information Technology & Electronic Engineering*, 20(4), 507-524, 2019.

[3] Kumar G, Kumar R. "Analysis of Arithmetic and Huffman Compression Techniques by Using DWT-DCT." *International Journal of Image, Graphics and Signal Processing*, 4, 63-70, 2021.

[4] Lu TC, Chang CY. "A Survey of VQ Codebook Generation". *Journal of Information Hiding and Multimedia Signal Processing,* 1(3), 190-203, 2010.

[5] Yang SB. "Constrained - storage multistage vector quantization based on genetic algorithms". *Pattern Recognition*, 41(2), 689–700, 2008.

[6] Chiranjeevi K, Jena UR. "Image compression based on vector quantization using cuckoo search optimization technique". *Ain Shams Engineering Journal,* 9(4), 1417–1431, 2018.

[7] Tsai CW, Tseng SP, Yang CS, Chiang MC. "Preaco: A fast ant colony optimization for codebook generation". *Applied Soft Computing*, 13(6), 3008–3020, 2013.

[8] Feng HM, Chen CY, Ye F. "Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression". *Expert Systems with Applications*, 32(1), 213–222, 2007.

[9] Horng MH. "Vector quantization using the firefly algorithm for image compression". *Expert Systems with Applications*, 39(1), 1078–1091, 2012.

[10] Guo JR, Wu CY, Huang ZL, Wang FJ, Huang MT. "Vector quantization image compression algorithm based on bat algorithm of adaptive separation search". *International Conference on Advanced Intelligent Systems and Informatics*, Cairo, Egypt, 11-13 December 2021.

[11] Geetha K, Anitha V, Elhoseny M, Kathiresan S, Shamsolmoali P, Selim MM. "An evolutionary lion optimization algorithm-based image compression technique for biomedical applications". *Expert Systems*, 38(1), Article e12508, 2021.

[12] Kumari GV, Rao GS, Rao BP. "Flower pollination-based K-means algorithm for medical image compression". *International Journal of Advanced Intelligent Paradigms*, 18(2), 171–192, 2021

[13] Rahebi J. "Vector quantization using whale optimization algorithm for digital image compression". *Multimedia Tools and Applications*, 81(14), 20077–20103, 2022.

[14] Althobaiti MM. Crow search algorithm based vector quantization approach for image compression in 6G enabled industrial internet of things environment. Editors: Gupta D, Ragab M, Mansour RF, Khamparia A, Khanna A. AI-enabled 6G networks and applications, 55–73, Wiley, 2023.

[15] Nag S. "Vector quantization using the improved differential evolution algorithm for image compression". *Genetic Programming and Evolvable Machines*, 20, 187–212, 2019.

[16] Ghadami R, Rahebi J. "Compression of images with a mathematical approach based on sine and cosine equations and vector quantization (VQ)". *Soft Computing*, 27(22), 17291–17311, 2023.

[17] Kilic I, Cetin M. "Improved adolescent identity search algorithm for block-based image compression". *Expert Systems with Applications*, 237, 121715, 2024.

[18] Kilic I. "A novel codebook generation by smart fruit fly algorithm based on exponential flight". *The International Arab Journal of Information Technology*, 20 (4), 584-591, 2023.

[19] Kilic I. "A Levy flight based BAT optimization algorithm for block-based image compression". *Technicki Glasnik – Technical Journal,* 16 (4), 477-483, 2022.

[20] Spall JC. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". *IEEE Transactions on Automatic Control*, 37(3), 332-341, 1992.

[21] Chen S, Mei T, Luo M, Yang X. "Identification of nonlinear system based on a new hybrid gradient-based PSO algorithm". *ICIA 2007 International Conference on Information Acquisition*, Jeju City, Korea, 8-11 July 2007.

[22] Kaveh A, Talatahari S. "A hybrid particle swarm and ant colony optimization for design of truss structures". *Asian Journal of Civil Engineering*, 9(4), 329–48, 2008.

[23] Plevris V, Papadrakakis M. "A hybrid particle swarm-gradient algorithm for global structural optimization". *Computer-Aided Civil and Infrastructure Engineering*, 26(1), 48-68, 2011.

[24] Cherki I, Chaker A, Djidar Z, Khalfallah N, Benzergua F. "A sequential hybridization of genetic algorithm and particle swarm optimization for the optimal reactive power flow". *Sustainability*, 11(14), 3862, 2019.

[25] Seyedpoor SM, Gholizadeh S. Talebian SR. "An efficient structural optimisation algorithm using a hybrid version of particle swarm optimisation with simultaneous perturbation stochastic approximation". *Civil Engineering and Environmental Systems*, 27(4), 295–313, 2010.

[26] Wessels S, van der Haar D. "Using particle swarm optimization with gradient descent for parameter learning in convolutional neural networks". *CIARP 25th Iberoamerican Congress*, Porto, Portugal, 2021.

[27] Pujari AK, Veeramachaneni SD. "Gradient based hybridization of PSO". *CSAI 2023 International Conference on Computer Science and Artificial Intelligence*, Beijing, China, 8 - 10 December 2023.

[28] Barroso ES, Parente JE, Cartaxo de Melo AM. “A hybrid PSO-GA algorithm for optimization of laminated composites”. *Structural and Multidisciplinary Optimization*, 55(6), 2111–2130, 2017.
[29] Parouha RP, Verma P. “Design and applications of an advanced hybrid meta-heuristic algorithm for optimization problems”. *Artificial Intelligence Review*, 54, 5931–6010, 2021.
[30] Shankar T, Shanmugavel S, Rajesh A. “Hybrid HSA and PSO algorithm for energy efficient cluster head selection in wireless sensor networks”. *Swarm and Evolutionary Computation*, 30, 1–10, 2016.
[31] Chegini SN, Bagheri A, Najafi F. “PSOSCALF: A new hybrid PSO based on Sine and Cosine Algorithm and Levy Flight for solving optimization problems”. *Applied Soft Computing*, 73, 697-726, 2018.
[32] Şenel FA, Gokce F, Yuksel AS, Yigit T. “A novel hybrid PSO–GWO algorithm for optimization problems”. *Engineering with Computers*, 35(4), 1359–1373, 2019.
[33] Kaya S, Karaçizmeli İH, Aydilek İB, Tenekeci ME, Gümüşçü A. “The effects of initial populations in the solution of flow shop scheduling problems by hybrid firefly and particle swarm optimization algorithms”. *Pamukkale University Journal of Engineering Science*, 26(1), 140–149, 2020.
[34] Qiao J, Wang G, Yang Z, Luo X, Chen J, Li K, Li P. “A hybrid particle swarm optimization algorithm for solving engineering problem”. *Scientific Reports*, 14(1), 8357, 2024.
[35] Kiranyaz S, Ince T, Gabbouj M. “Stochastic approximation driven particle swarm optimization with simultaneous perturbation – Who will guide the guide?”. *Applied Soft Computing*, 11(2), 2334–2347, 2011.
[36] Lloyd SP. “Least squares quantization in PCM”. *IEEE Transactions on information theory*, IT-28 (2), 129-137, 1982.
[37] Pena JM, Lozano JA. Larranage P. “An empirical comparison of four initialization methods for the K-Means algorithm”. *Pattern Recognition Letters*, 20, 1027-1040, 1999.
[38] Kennedy J, Eberhart R. “Particle swarm optimization”. *IEEE International Conference on Neural Networks*, Australia, 27 November - 1 December 1995.
[39] Shi Y, Eberhart R. “A modified particle swarm optimizer”. *IEEE Congress on Evolutionary Computation,* 4-9 May 1998.
[40] Gad AG. “Particle Swarm Optimization algorithm and its applications: A systematic review”. *Archives of Computational Methods in Engineering,* 29, 2531–2561, 2022.
[41] Spall JC. “Implementation of the simultaneous perturbation algorithm for stochastic optimization*”. IEEE Transactions on Aerospace and Electronics Systems*, 34(3), 817-823, 1998.