Engineering Science and Technology, an International Journal 19 (2016) 769–781

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: http://www.elsevier.com/locate/jestch

Full Length Article

# Fast vector quantization using a Bat algorithm for image compression

Chiranjeevi Karri *, Umaranjan Jena

*Department of Electronics and Telecommunication Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla 768018, Odisha, India*

ARTICLE INFO

*Article history:*
Received 29 July 2015
Received in revised form
6 November 2015
Accepted 6 November 2015
Available online 17 December 2015

*Keywords:*
Vector quantization
Linde–Buzo–Gray (LBG)
Particle swarm optimization (PSO)
Quantum particle swarm algorithm (QPSO)
Honey bee mating optimization (HBMO)
Firefly algorithm (FA)
Bat algorithm (BA)

ABSTRACT

Linde–Buzo–Gray (LBG), a traditional method of vector quantization (VQ) generates a local optimal codebook which results in lower PSNR value. The performance of vector quantization (VQ) depends on the appropriate codebook, so researchers proposed optimization techniques for global codebook generation. Particle swarm optimization (PSO) and Firefly algorithm (FA) generate an efficient codebook, but undergoes instability in convergence when particle velocity is high and non-availability of brighter fireflies in the search space respectively. In this paper, we propose a new algorithm called BA-LBG which uses Bat Algorithm on initial solution of LBG. It produces an efficient codebook with less computational time and results very good PSNR due to its automatic zooming feature using adjustable pulse emission rate and loudness of bats. From the results, we observed that BA-LBG has high PSNR compared to LBG, PSO-LBG, Quantum PSO-LBG, HBMO-LBG and FA-LBG, and its average convergence speed is 1.841 times faster than HBMO-LBG and FA-LBG but no significance difference with PSO.



## 1. Introduction

Image compression plays a major role in applications like internet browsing, medical science, navy applications, TV broadcasting and many more applications. Several techniques have been proposed by many researchers over the past decades for image compression. Vector Quantization (VQ) technique, performance is better than scalar quantization methods such as pulse code modulation (PCM), differential PCM (DPCM), Adaptive DPCM. Vector Quantization (VQ) [1,2] is basically a c-means clustering method widely used for image compression, pattern recognition [3,4], speech recognition [5], face detection [6] speech and image coding because of its excellent rate distortion performance. VQ is popular because it has simple decoding structure and can provide high compression ratio. Basically VQ is performed in three steps: 1. Vector formation: – division of image into non-overlapping blocks or vectors called input vectors, 2. Codebook generation: – a set of representative image blocks of the input blocks (vectors) is selected which is called a codebook and each representative image vector is called a codeword 3. Quantization: – here each input vector is approximated to a codeword in the codebook and corresponding index of this codeword is transmitted. VQ methods are classified into two categories, namely crisp and fuzzy [7]. Crisp VQ is based on hard decision making processes and appears to be sensitive in codebook initialization. The most representative algorithms of this category are the c-means. To improve the behavior of c-means, Linde et al. introduced the Linde–Buzo–Gray (LBG) algorithm, which begins with the smallest codebook size and gradually increase it using a splitting procedure [8]. The performance of the LBG is improved by embedding special functions called utility measures in the learning process. Fuzzy VQ is carried out in terms of fuzzy cluster analysis. The most representative algorithms of this category is the fuzzy c-means. The fuzzy c-means assume that each training vector belongs to multiple clusters with different participation degrees (i.e. Membership degrees). Therefore, the learning is a soft decision making process [9]. LBG Algorithm undergoes local optimum problem [10]. So Patane and Russo proposed a clustering algorithm called enhanced LBG (ELBG) [11]. They used the concept of utility of a codeword to overcome the local optimal problem of LBG by shifting the lower utility codewords to the one with higher utility. Recently, the evolutionary optimization algorithms had been developed to design the codebook for improving the results of LGB algorithm. Rajpoot, Hussain, Saleem and Qureshi designed a codebook by using an ant colony system (ACS) algorithm [12]. In this algorithm, codebook is optimized by representing the vector coefficients in a bidirectional graph, followed by defining a suitable mechanism of depositing pheromone on the edges of the graph. Tsaia, Tsengb, Yangc, and Chiangb proposed a fast ant colony optimization for codebook generation by observing the redundant calculations [13]. In addition, particle swarm optimization (PSO) vector quantization [14] outperforms LBG algorithm which is based on updating the global best (gbest) and local best (lbest) solution. Feng, Chen, and Fun showed that Evolutionary fuzzy particle swarm optimization algorithm [15] has better global and robust performances than LBG learning

* *Corresponding author.* Tel.: +91 9441177331, fax: (0663)2430204.
*E-mail address:* chiru404@gmail.com (C. Karri).
Peer review under responsibility of Karabuk University.

http://dx.doi.org/10.1016/j.jestch.2015.11.003
2215-0986/

algorithms. Quantum particle swarm algorithm (QPSO) was proposed by Wang, Feng, Huang, Zhou, and Liang to solve the 0–1 knapsack problem [16]. The QPSO performance is better than PSO as it computes the local point from the *pbest* and *gbest* for each particle and updates the position of the particle by choosing appropriate parameters u and z.

Horng and Jiang applied honey bee mating optimization algorithm for Vector quantization [17]. HBMO has high quality reconstructed image and better codebook with small distortion compared to PSO-LBG, QPSO-LBG and LBG algorithm. Horng applied a firefly algorithm (FA) to design a codebook for vector quantization [18]. The firefly algorithm has become an increasingly important tool of Swarm Intelligence that has been applied in almost all areas of optimization, as well as engineering problems. Firefly algorithm is encouraged by social activities of fireflies and the occurrence of bioluminescent communication. A Firefly with lighter intensity value move towards the brighter intensity firefly, and if there is no brighter firefly then it moves randomly. Chang, Chiang, Yeh proposed a tree structured vector quantization for fast codebook design with the help of employing the triangle inequality to achieve efficient pruning of impossible codewords [19]. Chen, Hwang and Tsou proposed a fast codebook search algorithm based on triangular inequality estimation [20]. Kekre, Sarode, Sange, Natu and Natu proposed a fast codebook search algorithm with different codebook sizes in 8, 16, 32, 64, 128 and 256 based on the kekre's fast codebook generation algorithm [21]. Kekre, Sarode, Thepade and Vaishali proposed a Kekre's fast codebook generation in VQ with various color spaces for colorization of grayscale images [22]. Yang, RuiXia, Wang, and Jiao proposed an Evolutionary clustering based vector quantization for image compression based on One-step gradient descent genetic algorithm (OSGD-GA). OSGD-GA is designed for optimizing the codebooks of the low-frequency wavelet coefficient by defining the importance of each coefficient and utilizing fuzzy membership to address the automatic clustering [23]. Multivariate vector quantization (MVQ) approach is useful for compression of hyper spectral imagery (HSI) based on a linear combination of two codewords from the codebook, and the indexed maps and their corresponding coefficients are separately coded and compressed [24]. Contextual vector quantization (CVQ) compresses the medical ultrasound (US) images [25]. Contextual region is defined as a region containing the most important information and must be encoded without considerable quality loss. Attempts are made to encode this region with high priority and high resolution CVQ algorithm. Huanga, Wanga and Chen proposed a dynamic learning vector–scalar quantization for compression of ECG image by forming DWT coefficients into tree vector (TV) and on which vector–scalar quantization is performed [26]. Tripathy, Dash and Tripathy proposed a network layout optimization based on dynamic programming using optimization techniques [27]. Tsolakis et al. (in 2012) proposed a Fuzzy vector quantization for image compression based on competitive agglomeration and a novel codeword migration strategy [28]. George et al. proposed an improved batch fuzzy learning vector quantization for image compression [29]. Tsekouras (in 2005) proposed a fuzzy vector quantization by assigning an input vector to more than one codeword based on the crisp relation [30]. Tsolakis et al. developed a fast fuzzy vector quantization for gray scale image compression by combining three different learning modules. Those are: 1. Remove codewords that are affected by a specific training pattern; 2. Reduce the number of training patterns; 3. Codewords of small clusters have moved to the neighborhood of large ones [31].

Bat algorithm (BA) is a nature inspired Metaheuristic algorithm developed by Yang in 2010 [32]. There is functional similarity between bat algorithm and radio detection and ranging (RADAR). The radar works based on examination of reflected signals/echo signal from the object. Similarly, the basic idea behind Bat algorithm is an echolocation feature of micro bats. Bat algorithm is based on the echolocation behavior of micro bats with varying pulse rates of emission and loudness. The bat emits some sounds with different pulse rate and loudness. These sound signals are reflected back from objects called echo signals. With these echo signals, bats can determine the size of the object and distance to object, speed of the object and even their texture in fractions of a second because of their sophisticated sense of hearing. Frequency tuning, automatic zooming and parameter control features helps the Bat algorithm to be efficient and speedy. Also Bat algorithm is simple and flexible. A detailed comparison of bat algorithm with LBG, PSO, QPSO, HBMO and FA for image compression with vector quantization is given in Tables 5–12 and Figs. 6–15. From comparison, we conclude clearly that BA has advantages over other algorithms. Along with optimization problems [33], Bat algorithm can also be applied to classification, clustering, data mining [34], image matching [35], Fuzzy logic [36], and parameter estimation [37] problems.

This paper is organized into five sections including the introduction. In section 2, recent methods of codebook design are discussed along with their algorithms. The proposed method of BA-LBG algorithm is presented with the procedures in section 3. The results and discussions are given in section 4. Finally, the conclusion is given in section 5.

## 2. Recent methods of codebook design for VQ

As discussed in section 1, the major important technique for image compression is Vector Quantization (VQ), which is to be optimized. The Vector Quantization (VQ) is one of the block coding technique for image compression. Codebook design is an important task in the design of VQ that minimizes the distortion between reconstructed image and original image with less computational time. Fig. 1 shows the encoding and decoding process of vector quantization. The image (size $N \times N$ ), to be vector quantized, is subdivided into $N_b$ $(\frac{N}{n} \times \frac{N}{n})$ blocks with size $n \times n$ pixels. These divided image blocks or training vectors of size n × n pixels are represented with $X_i$ $(i = 1, 2, 3, \ldots, N_b)$. The Codebook has a set of code words, where $C_i$ $(i = 1 \ldots NC)$ is the $i^{th}$ codeword. The total number of codewords in Codebook is $N_c$. Each subdivided image vector is (approximated) represented by the index of a codeword based on the minimum Euclidean distance between the corresponding vector and code words. The encoded results from the index table. During the decoding procedure, the receiver uses the same codebook to translate the index back to its corresponding codeword for reconstructing the image. The distortion $D$ between training vectors and the codebook is given as

$$D = \frac{1}{N_c} \sum_{j=1}^{N_c} \sum_{i=1}^{N_b} u_{ij} \cdot \|X_i - C_j\|^2 \tag{1}$$

Subject to the following constraints:

$$D = \sum_{j=1}^{N_c} u_{ij} = 1 \quad i \in \{1, 2, \ldots, N_b\} \tag{2}$$

$u_{ij}$ is one if $X_i$ is in the $j^{th}$ cluster, otherwise zero.

Two necessary conditions exist for an optimal vector quantizer.

(1) The partition $R_j$, $j = 1, \ldots, N_c$ must satisfy

$$R_j \supset \{x \varepsilon X : d(c, C_j) < d(x, C_k), \quad \forall k \neq j\} \tag{3}$$

(2) The codeword $C_j$ must be given by the centroid of $R_j$:

$$C_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i \quad x_i \in R_j \tag{4}$$

where $N_j$ is the total number of vectors belonging to $R_j$.

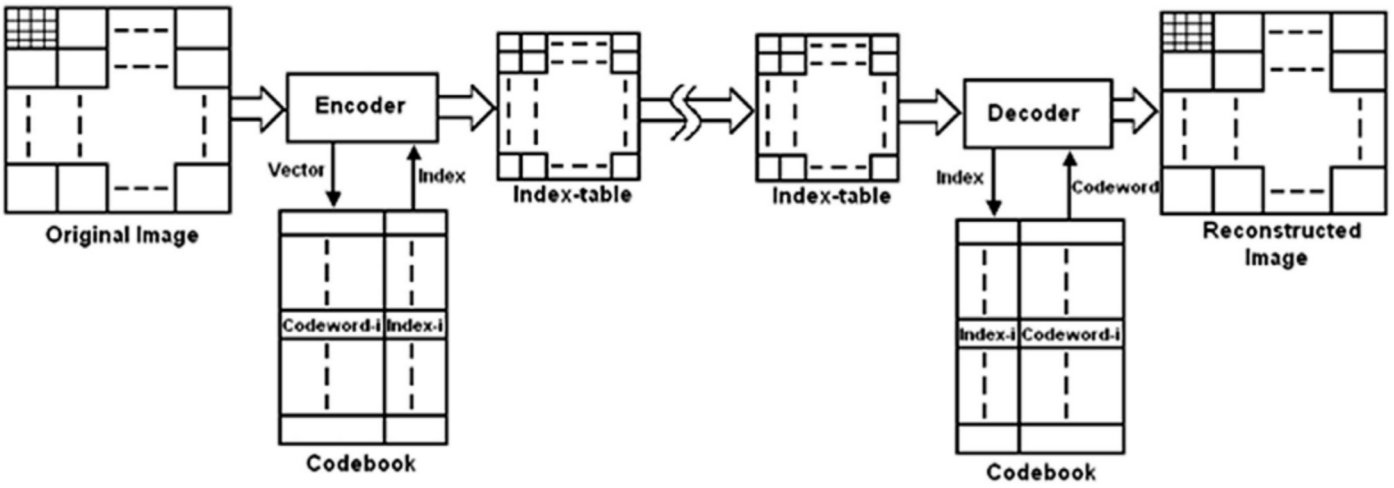



**Fig. 1.** Encoding and decoding process of vector quantization.

### 2.1. Generalized LBG vector quantization algorithm

The most commonly used methods in VQ are the Generalized Lloyd Algorithm (GLA) which is also called Linde–Buzo–Gray (LBG) algorithm. The algorithm is as follows:

Step 1: Begin with initial codebook $C_1$ of size $N$. Let the iteration counter be $m = 1$ and the initial distortion $D_1 = \infty$
Step2: Using codebook $C_m = \{Y_i\}$, partition the training set into cluster sets $R_i$ using the nearest neighbor condition.
Step 3: Once the mapping of all the input vectors to the initial code vectors is made, compute the centroids of the partition region found in step 2. This gives an improved codebook $C_{m+1}$.
Step 4: Calculate the average distortion $D_{m+1}$. If $D_m - D_{m+1} < T$ then stops, otherwise $m = m + 1$ and repeat steps 2–4.

The distortion becomes smaller after recursively executing the LBG algorithm. Actually, the LBG algorithm can guarantee that the distortion will not increase from iteration to the next iteration. However, it cannot guarantee the resulting codebook will become the optimum one, and the initial condition will significantly influence the results. Therefore, in the LBG algorithm, we should pay more attention to the choice of the initial codebook.

### 2.2. PSO-LBG vector quantization algorithm

The PSO was proposed by Kennedy and Eberhart in the year 1995 [38]. It is based on social behavior of bird flocking or fish schooling. There are two categories of PSO models: *gbest* and *lbest* models. PSO *gbest* model was used by Chen, Yang and Gou [39] to design a codebook for vector quantization by initializing the result of a LBG algorithm as *gbest* particle so that it will increase the speed of convergence of PSO. In PSO, particles (or codebooks) alter their values based on their previous experience and the best experience of the swarm to generate a best codebook. The structure of codebook for optimization techniques with size $N_c$ and length $N_b$ is shown in Fig. 2. Here, codebook is assumed as particles. The PSO algorithm follows:

Step 1: Run the LBG algorithm; assign it as global best codebook (*gbest*).
Step 2: Initialize rest codebooks with random numbers and their corresponding velocities.
Step 3: Find out fitness values by Eq. (5) for each codebook.

$$\text{Fitness}(C) = \frac{1}{D(C)} = \frac{N_b}{\sum_{j=1}^{N_c}\sum_{i=1}^{N_b} u_{ij} \cdot \|X_i - C_j\|^2} \tag{5}$$

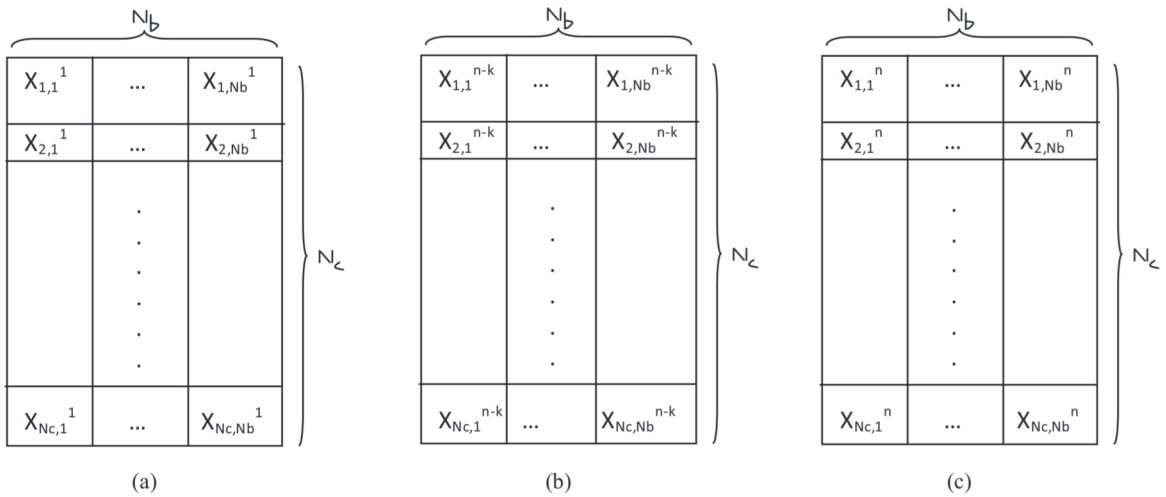



**Fig. 2.** The structures of codebook in Swarm optimization techniques (a) The first codebook ($N_c \times N_b$); (b) the $(n-k)^{th}$, codebook ($N_c \times N_b$) (c) The $n^{th}$ codebook ($N_c \times N_b$).

Step 4: If codebook new fitness value is better than old fitness (*pbset*), then assign its corresponding new fitness as *pbest*.
Step 5: Select the highest fitness value among all the codebooks, and if it is better than *gbest*, then replace *gbest* with highest fitness value.
Step 6: Update the velocities by Eq. (6) and update each particle to a new position by Eq. (7) and return to Step 3.

$$v_{ik}^{n+1} = v_{ik}^{n} + c_1 r_1^n \left(\text{pbest}_{ik}^n - \text{X}_{ik}^n\right) + \text{c}_2\text{r}_2^n \left(\text{gbest}_k^n - \text{X}_{ik}^n\right) \tag{6}$$

$$X_{ik}^{n+1} = \text{X}_{ik}^n + \text{v}_{ik}^{n+1} \tag{7}$$

Where $k$ is the number of solutions, $i$ is position of the particle, $c_1$, $c_2$ are cognitive and social learning rates respectively. $r_1$ and $r_2$ are random numbers.
Step 7: Until a stopping criterion is satisfied (Maximum iteration) repeat Step 3–7.

### 2.3. QPSO-LBG vector quantization algorithm

Unlike PSO, the QPSO computes the local point $P_i$ [8] from the pbest and gbest for $i^{th}$ codebook according to Equation (8).

$$p_i = r_1 pbest_i + r_2 gbest_i / r_1 + r_2 \tag{8}$$

Furthermore, two parameters, $u$ and $z$, are defined to update the position of the particle associated with the local point. To sum up, the three parameters are used to update the particle. Here, codebook is assumed as particle. The detail algorithm

Step 1: Run the LBG algorithm; assign it as global best codebook (*gbest*) and initialize the rest of codebooks and associated velocities randomly.
Step 2: Find out fitness values by Eq. (5) for each codebook.
Step 3: If any codebook's new fitness value is better than its old fitness value (*pbset*), then assign the corresponding new fitness at its *best*.
Step 4: Select the highest fitness value among all the particles, and if it is better than *gbest*, then replace *gbest* with highest fitness value.
Step 5: Select *r1*, *r2*, and *u* randomly in the range 0 to 1, further compute the local point $p_i$ based on Eq. (8).
Step 6: Update each element of codebook $X_i$ as follows:

$$L_i = z\left|X_i - p_i\right| \tag{9}$$

$$\text{If } \text{u} > 0.5 \quad X_i(\text{t}+1) = \text{p}_i - L_i \cdot \ln(1/\text{u}) \tag{10}$$

$$\text{else } X_i(t+1) = p_i + L_i \times \ln(1/u)$$

Where $z$ is non-negative constant and is less than $1/\ln\sqrt{2}$ and t is the current iteration time.
Step 7: Repeat steps (3) to (7) until stop criteria is satisfied.

### 2.4. HBMO-LBG vector quantization algorithm

Many algorithms are proposed by the researchers based on the behavior of honey bees [40]. These algorithms are mainly divided in two categories according to their behavior in the nature, the foraging behavior and the mating behavior. Based on the foraging behavior of the bees, the Artificial Bee Colony (ABC) Algorithm was proposed by Karaboga and Basturk [41], Honey Bee Mate optimization (HBMO) is based on mating process of honey bees [42]. Here, codebooks are assumed as Bees. The detailed algorithm for vector quantization is as follows:

Step 1: Assign the codebook of LBG algorithm as a queen $Q$ and generate other codebooks (drones) randomly. Initialize $\alpha$, queen's spermatheca, energy and speed.
Step 2: Calculate the fitness of all the drones by Eq. (5) and assign largest as *Dbest*, If the *Dbest* is better than the queen $Q$ fitness then queen $Q$ is replaced by *Dbest*.
Step 3: Queen selects a drone for mating if the drone passes the probabilistic condition given by Eq. (11) and then adds sperm of the drone in the spermatheca.

$$Prob(D) = e^{\left[\frac{-\Delta(f)}{Speed(t)}\right]} \tag{11}$$

Step 4: The new brood is generated from the queen $Q$ and the sperm by using Eqs. (12)–(14)

$$brood = Q \pm \beta \times (sperm - Q) \tag{12}$$

$$Speed(t+1) = \alpha \times Speed(t) \tag{13}$$

$$energy(t+1) = \alpha \times energy(t) \tag{14}$$

where $\beta$ and $\alpha$ are random numbers ranged between 0 and 1.
Step 5: Select a sperm from the spermatheca and generate a brood by applying a crossover operator between the queen and the selected drone.
Step 6: If the brood's fitness is better than the queen's fittest then replacing the queen with the brood else if the brood's fitness is better than one of the drone's fitness then replace the drone with the brood.
Step 7: Repeat step 3 to 7 until maximum iterations.

### 2.5. Firefly vector quantization algorithm

Honey Bee Mate optimization (HBMO) technique's performance is based on many parameters, i.e. many interdependent parameters are required to tune for efficient codebook design, and it's a difficult task for the researchers. So, the Firefly algorithm (FA) was introduced by Yang (in 2008) [43]. The FA is inspired by the flashing pattern and characteristics of fireflies. The brightness of a firefly is equal to objective function value. The lighter firefly (lower fitness value) moves towards brighter firefly (higher fitness value). Here, codebooks are assumed as fireflies. FF-LBG algorithm is faster than the PSO, QPSO and HBMO algorithms and the reconstructed images get higher quality than those generated from the LBG, PSO and QPSO, but it is no significant superiority to the HBMO algorithm. The detailed FA algorithm is given below.

Step 1: Run the LBG algorithm once and assign it as brighter codebook.
Step 2: Initialize $\alpha$, $\beta$ and $\gamma$ parameters. Initialize rest codebooks with random numbers.
Step 3: Find out fitness values by Eq. (5) of each codebook.
Step 4: Randomly select a codebook and record its fitness value. If there is a codebook with higher fitness value, then it moves towards the brighter codebook (highest fitness value) based on the Eqs. (15)–(17).

$$\text{Euclidean distance } r_{ij} = \|X_I - X_J\| = \sqrt{\sum_{k=1}^{N_c}\sum_{h=1}^{L}\left(X_{i,k}^h - X_{j,k}^h\right)^2} \tag{15}$$

Here, $X_i$ is randomly selected codebook, $X_j$ is brighter codebook

$$\beta = \beta_0 e^{-\gamma_{i,j}} \tag{16}$$

$$X_{j,k}^h = (1-\beta)X_{i,k}^h + \beta X_{j,k}^h + u_{j,k}^h \tag{17}$$

where $u_{ij}$ is random number between 0 to1, $k = 1,2,\ldots,N_c$, $h = 1,2,\ldots,L$.

Step 5: If selected firefly doesn't find brighter fireflies in search space, then it moves randomly with the following equation.

$$X_{i,k}^{h} = X_{i,k}^{h} + u_{j,k}^{h} \quad k = 1,2,\ldots,N_c, \quad h = 1,2,\ldots,L \tag{18}$$

Step 6: Repeat step (3) to (5) until one of the termination criteria is reached.

## 3. Proposed BA-LBG vector quantization algorithm

PSO and Quantum PSO generate an efficient codebook but undergo instability in convergence when particle velocity is of very high value [44]. HBMO algorithm provides a near optimum codebook in a limited runtime period. Firefly algorithm (FA) was developed to generate near global codebook, but it experienced a problem when no such significant brighter fireflies are found in the search space [45]. So we developed a Bat algorithm (BA) that gives a global codebook with minimum number of iterations and with two tuning parameters (loudness and pulse rate). It is a nature inspired metaheuristic algorithm developed by Yang in 2010. Bat algorithm works with the three assumptions: 1) All bats use echolocation to sense distance, and they also 'know' the difference between food (called prey) and background barriers in some magical way; 2. Bats fly randomly with velocity $V_i$ at position $X_i$ with a fixed frequency $Q_{min}$, varying wavelength $\lambda$ and loudness $A_0$ in search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target; 3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) $A_0$ to a minimum constant value $A_{min}$. Intensification (local search/local optimum) of the algorithm is attained with pulse rate and diversification (global search/local optimum) is attained with loudness parameter [46]. The structure of the codebook is as shown in Fig. 2. Here, codebooks are assumed as Bats. Bat algorithm works with the calculation of distance between bat and object, the position of the bat and the frequency of a bat. To calculate the frequency of bat, let us consider the codebook $C$ and the number of bats are $\{B1, B2,\ldots,Bk,\ldots,Bn\}$. Each codebook is considered as one bat. Let us denote the frequency of a sound as $Q$ for a bat $B$. The frequency $Q_k$ of bat $B_k$ can be achieved by Eq (19). Where $R_1$ is the pulse rate used to control the frequency $Q_k$ of bat $B_k$, and when it reaches near or far from the object, the value of $R_1$ is auto adjusted in each iteration by Eq. (23).

$$Q_k = R_1 \sum_{i=1}^{m} (C_{ki})/m \tag{19}$$

The Distance $S$ of the object $z$ from bat $B_k$ is calculated by multiplying $C_k$ with some random weight value multiplied by $Q_k$ for each object $z$.

$$S_{objectz} = Q_k * C_k * w \tag{20}$$

In Eq (20), $w$ is a step size of random walk. After calculating the error by Eq (21), the bat position can be changed by Eq (22).

$$E_k = S_{objectz} - 1 \tag{21}$$

$$X_k = X_k + E_k \tag{22}$$

When bat starts flying, it assumes that the position is initialized to zero. Its position keeps on changing when it reaches nearer to the object. As closer it moves to the pray, error $E_k$ and position $X_k$ reduces to zero. Update frequency $Q$ and step size for random walk $w$ after change of position of bat. As the bat reaches nearer to its object the frequency starts reducing. This can be achieved by controlling the value of $R_1$ in Eq (19) by using Eq (23). $R_2$ is a constant treated as the bat learning parameter

$$R_1 = Q_k + R_2 * E_k^2 * X_k \tag{23}$$

$$w = w + 2 * u * E_k \tag{24}$$

Where $u$ is random number.

The flow chart of Bat algorithm is as shown in Appendix. The detailed Bat algorithm is as follows:

Step 1: (Initialize the bat/codebook & parameters): Initialize the number of codebooks $N$, loudness $A$, velocity $V$, pulse rate $R$, minimum frequency $Qmin$ and maximum frequency $Qmax$. Codebook of LBG algorithm is assigned as one of initial codebook $X_1$ and other codebooks $X_i$, $(i = 2, 3, \ldots, N-1)$ are selected randomly.

Step 2: (Find the best codebook): Calculate the fitness of all codebooks using Eq. (5), and best fitness codebook is assigned $Xbest$.

Step 3: (Automatic zooming of codebooks towards $Xbest$): Zoom each element of each codebook as in Eq. (27) by adjusting frequency in Eq. (25) and velocities in Eq. (26).

$$\text{Frequency update: } Q_i(t+1) = Qmax(t) + (Qmin(t) - Qmax(t)) * R \tag{25}$$

$$\text{Distance update: } v_i(t+1) = v_i(t) + (X_i - Xbest) * Q_i(t+1) \tag{26}$$

$$\text{Position update: } X_i(t+1) = X_i(t) + v_i(t+1) \tag{27}$$

Step 4: (Selection of step size of random walks): If the generated random number is greater than pulse rate '$R$' then move the codebooks around the selected best codebook based on Eq. (28).

$$X_i(t+1) = Xbest(t) + w * R \tag{28}$$

Here, $w$ is the step size for random walk which is selected randomly within 0 to 1.

Step 5: (Generate a new codebook by flying randomly): Generate a random number, if it is less than loudness and new codebook fitness is better than the old codebook, accept the new codebook.

Step 6: Rank the bats and find the current best $Xbest$.

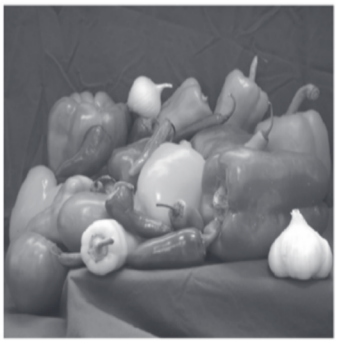
Step 7: Repeat step (2) to (6) until maximum iterations.

## 4. Simulation results and discussion

In section three, we discussed different optimization techniques for vector quantization. In this section, we carried out computational experiments to show the effectiveness of our newly proposed BA-LBG method. The empirical experiments are performed on Windows XP PC with an Intel (R) Core (TM) i5-2540 Machine with 2.60 GHz CPU, and 2.94 GB of RAM. Moreover, all the programs are written and compiled on MATLAB version 7.9.0 (R2009b). We have chosen five tested images: "LENA", "BABOON", "PEPPER", "BARB" and "GOLDHILL" for comparison of bat algorithm with other algorithms, as shown in Figs. 3(a), 4(b), 4(c), 4(d), 4(e) respectively. All the images are grayscale images of size $512 \times 512$ pixels. Among all the images, pepper is ".png" format and remaining images are "jpg" format. The images are compressed with BA-LBG, FA-LBG, HBMO-LBG, QPSO-LBG, PSO-LBG and generalized Lloyd algorithm (LBG). As discussed in section 2, the image to be compressed is subdivided into non-overlapping images with size $4 \times 4$ pixels. Each subdivided image is treated as a training vector

(a). LENA image

(b). BABOON image



(c). PEPPERS image

(d). BARB image

(e). GOLDHILL image

**Fig. 3.** The five test images: (a) LENA, (b) BABOON, (c) PEPPERS, (d) BARB and (e) GOLDHILL.

with $(4\times4)$ 16 dimensions. So there are 16384 $(\frac{512}{4}\times\frac{512}{4})$ training vectors to be encoded.

The parameters used for comparison of proposed bat algorithm with others are bitrate/bits per pixel, Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE) as given in Eqs. (29), (30) and (31) respectively. PSNR and fitness values are calculated for all the images with different codebook sizes of 8, 16, 32, 64, 128, 256, 512 and 1024. We use bpp (bit per pixel) to evaluate the data size of the compressed image for various codebook sizes of 8, 16, 32, 64, 128, 256, 512 and 1024. We then use the PSNR (peak signal-to-noise ratio) to evaluate the quality of the reconstructed image.

$$\text{bpp} = \frac{\log_2 N_c}{k} \tag{29}$$

Where $N_c$ is codebook size and $k$ is non-overlapping image block size $(4\times4)$

$$PSNR = 10\times10\log\left(\frac{255^2}{MSE}\right)(\text{dB}) \tag{30}$$

Where (MSE) which is given by the equation

$$MSE = \frac{1}{M\times N}\sum_{i}^{M}\sum_{j}^{N}\left\{f(\text{i},\text{j}) - \bar{f}(\text{i},\text{j})\right\}^2 \tag{31}$$

Where $M\times N$ is the size of the image, $i$ and $j$ represents the pixel coordinates of original and decompressed images. In our experiment, we have taken $N = M$, *that means* a square image. $f(\text{i},\text{j})$ is an original image and $\bar{f}(\text{i},\text{j})$ is a reconstructed image of size 512 by 512 each. The Bat algorithm parameter values used for simulating the images are chosen based on the maximum of average PSNR of experiments being performed (three times in our case) and are shown in Fig. 4. From experimental observations, we selected Loudness = 0.7 and Pulse rate = 0.2 parameters values for later experiments. But the step-size of random walks = 0.19 is selected randomly. BA can use echolocation or frequency tuning for problem solution, but experimentally it is observed that for vector quantization BA works well with frequency tuning instead of echolocation. The parameters used for simulating PSO-LBG, HBMO-LBG, FA-LBG and BA-LBG are tabulated in Tables 1–4. All

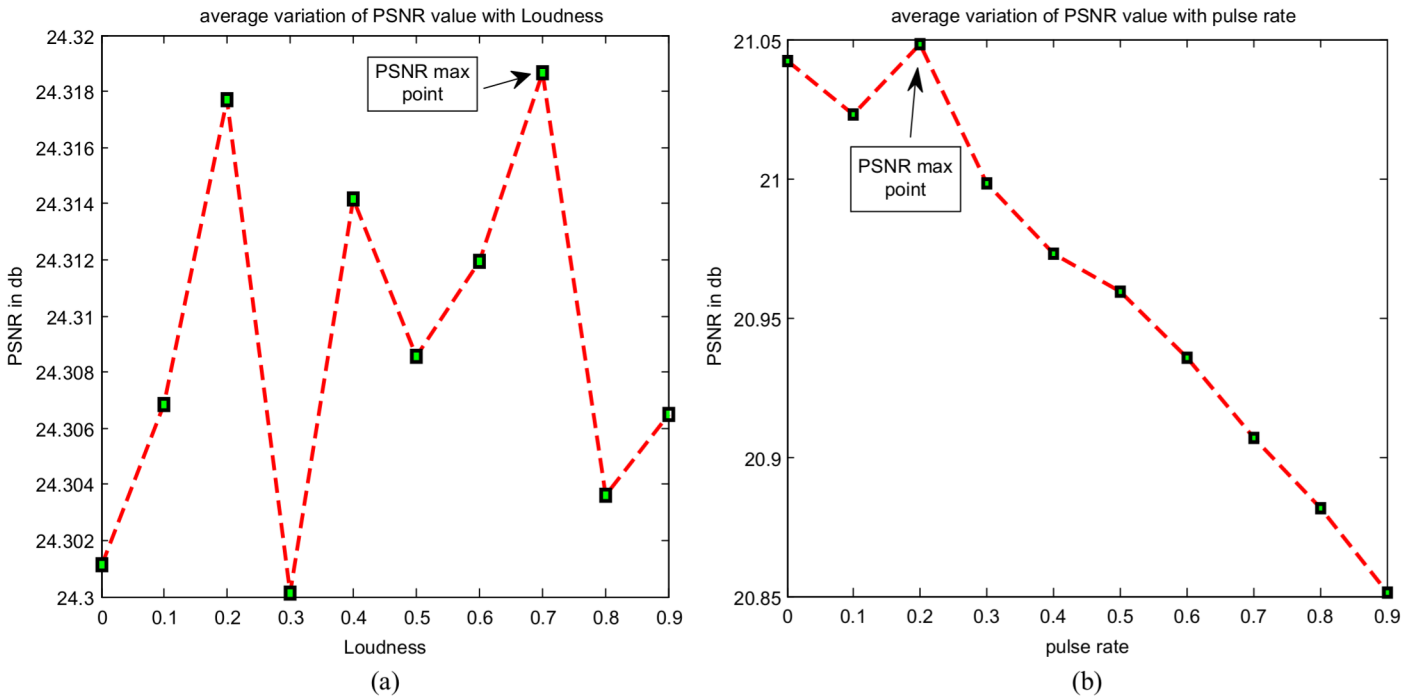



**Fig. 4.** Average PSNR of LENA image being performed 5 times for selection of parameter (a) loudness (b) pulse rate.

the experiments are performed three times. To understand the performance of proposed method, we have drawn the graphs of average peak signal to noise ratio (PSNR) of each method against bitrate (BR). Figs. 5–9 shows the average peak signal to noise ratio of different tested images against bitrate. Experimentally, it is shown that bat algorithm improves the PSNR values by around 0.2 dB at low bit rates and 0.5 dB at higher bit rates. Bat algorithm PSNR is better even for pepper image which has low spatial frequency components. Experimentally, it is observed from the graphs that for different codebook sizes, bat algorithm PSNR value is better than LBG, PSO-LBG, QPSO-LBG, HBMO-LBG and FA-LBG. These graphs reveal that for all algorithms PSNR value is better than the LBG algorithm.

Tables 5–12 shows the average computation time or convergence time of different algorithms with different bitrates. Horng and Jiang in his paper [18] simulated the five algorithms in 'C++6.0' with windows XP operating systems, number of codebooks/solutions 100 and iterations 50. Whereas we simulated the six algorithms in

**Table 1**
The parameters used in the QPSO-LBG algorithm.

| Parameter | Explanation | Value |
|---|---|---|
| S | Number of particles | 30 |
| iter | Number of iterations | 20 |
| l | Random numbers | [0,1] |
| r1 | Random number | [0,1] |
| r2 | Random number | [0,1] |
| z | Non-negative constant | 0.9 |
| Lb | Lower limit | 0 |
| Ub | Upper limit | 255 |

**Table 2**
The parameters used in the HBMO-LBG algorithm.

| Parameter | Explanation | Value |
|---|---|---|
| | Number of queens | 1 |
| M | Number of drones | 30 |
| c | Number of threshold | 2 |
| L | The grayscale of image | 255 |
| iter | Number of iterations | 20 |
| a | Speed reduction schema | 0.98 |
| $n_{sperm}$ | Capacity of spermatheca | 50 |
| S (0) | Speed of queen at first of flight | [0.5, 1] |
| Pc | The breeding ratio | 0.8 |
| Pm | Mutation ratio | 0.01 |
| e | Mutation variation | 0.5 |

**Table 3**
The parameters used in the FA-LBG algorithm.

| Parameter | Explanation | Value |
|---|---|---|
| n | Population size | 30 |
| itrer | Iterations | 20 |
| α | Alpha | 0.01 |
| β0 | Beta minimum | 1 |
| γ | Gamma | 1 |
| Ub | Upper limit | 255 |
| Lb | Lower limit | 0 |

**Table 4**
The parameters used in the BA-LBG algorithm.

| Parameter | Explanation | Value |
|---|---|---|
| n | Population size | 30 |
| iter | Iterations | 20 |
| A | Loudness | 0.7 |
| R | Pulse rate | 0.2 |
| $Q_{min}$ | Frequency minimum | 20 |
| $Q_{max}$ | Frequency maximum | 38 |
| Lb | Lower limit | 0 |
| Ub | Upper limit | 255 |
| Q | Initial frequency | 0 |
| V | Initial velocity | 0 |
| W | Step sizes of random walk | 0.19 |

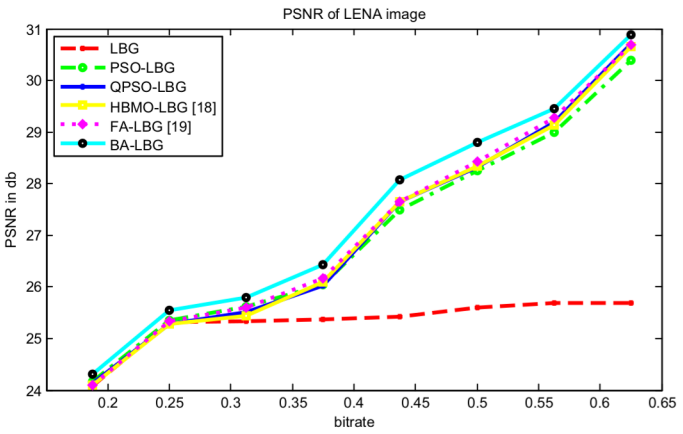



**Fig. 5.** The average PSNR of six vector quantization methods for LENA image.

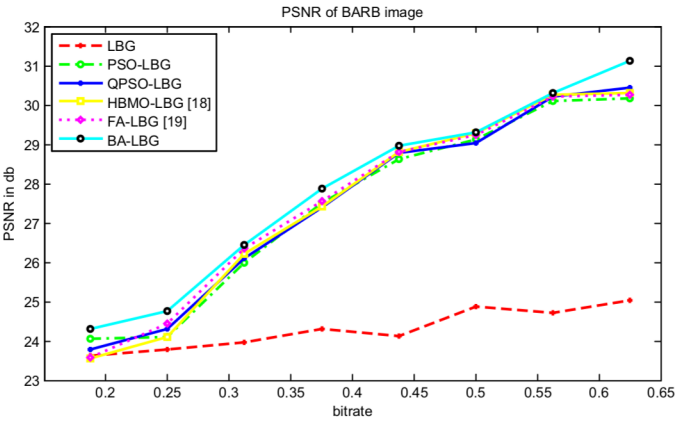



**Fig. 8.** The average PSNR of six vector quantization methods for BARB image.

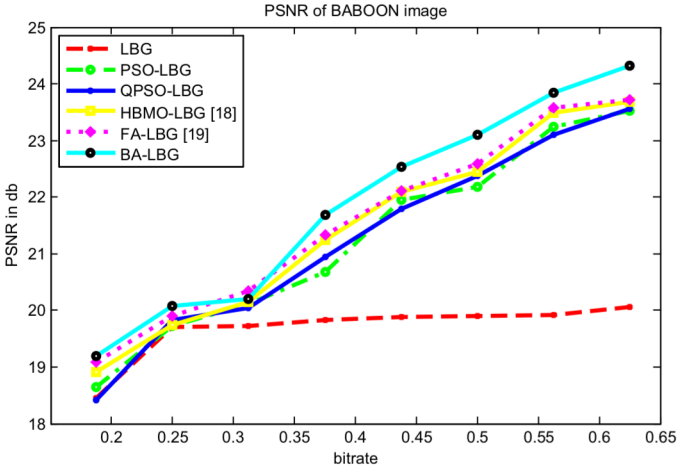



**Fig. 6.** The average PSNR of six vector quantization methods for BABOON image.

MATLAB with codebooks 30 and iterations 20, so there is some dissimilarity of average computational time between proposed BA-LBG and FA-LBG. The C++ computation time is 10 to 100 times faster than MATLAB when numbers of instruction lines are more than 50 to 100 [47]. In similar Ming-Huwi Horng in his paper [19] simulated the five algorithms in MATLAB with 100 iterations. As we run all the algorithms in MATLAB with 30 solutions and 20 iterations, there is small dissimilarity in computational time between proposed BA-LBG and FA-LBG. From the observations of table LBG algorithm, computational time is very less compared to all other algorithms, but has lesser PSNR and bad reconstructed image. The average computation time of Bat algorithm is around 1.841 times faster than the firefly algorithm and honey bee mate optimization techniques. Automatic zooming feature of BA helps to reduce the convergence time of codebook design for vector quantization. As the convergence speed of Bat algorithm is fast, it is applied to design a fast codebook for vector quantization. So we call bat algorithm as one of the fastest vector quantization codebook search algorithm. The average computation time of bat algorithm is almost similar to that of PSO and QPSO. The normal fitness values of the five experiment images by using the six vector quantization algorithms are shown in Figs. 10–14. These investigation's confirmed that the fitness of the five test images using the BA-LBG algorithm is higher than the LBG, PSO-LBG, QPSO-LBG, HBMO-LBG and FA-LBG algorithm. Figs. 15–19 shows the six vector quantization methods, decompressed five test images with codebook size 256 and block size 16. It is observed that the reconstructed image quality of the BA-LBG algorithm has superior quality than the LBG, PSO-LBG, QPSO-LBG, HBMO-LBG and FA-LBG algorithms.

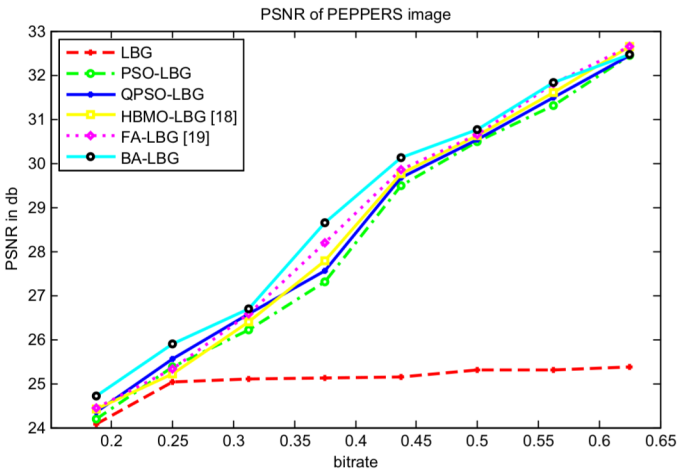



**Fig. 7.** The average PSNR of six vector quantization methods for PEPPER image.

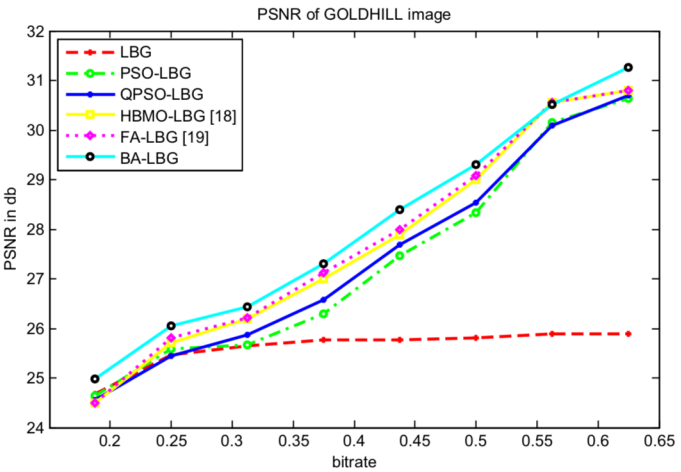



**Fig. 9.** The average PSNR of six vector quantization methods for GOLDHILL image.

**Table 5**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.1875.

| Codebook size: **8** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 3.371542 | 254.357919 | 261.299374 | 890.254347 | 877.123889 | 323.150506 |
| PEPPER | 3.416869 | 247.181509 | 252.652972 | 676.344334 | 660.141175 | 276.213142 |
| BABOON | 4.313500 | 322.996476 | 326.493443 | 723.897865 | 705.210035 | 271.339974 |
| GOLDHILL | 3.622867 | 247.211833 | 346.986628 | 681.978545 | 661.281546 | 298.879569 |
| BARB | 3.843362 | 257.520535 | 268.403187 | 654.976675 | 631.435366 | 317.623761 |
| Average | 3.713628 | 265.8536544 | 291.1671208 | 725.4903532 | 707.0384022 | 297.4413904 |

**Table 6**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.2500.

| Codebook size: **16** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 3.405184 | 252.001592 | 267.208254 | 528.995495 | 507.183026 | 255.030922 |
| PEPPER | 4.575627 | 250.411978 | 253.430517 | 567.656548 | 534.304628 | 323.788582 |
| BABOON | 4.537825 | 321.669194 | 333.844743 | 952.324245 | 943.362724 | 335.664621 |
| GOLDHILL | 4.907262 | 318.004351 | 376.746590 | 589.097844 | 574.986090 | 261.008145 |
| BARB | 4.391757 | 264.414665 | 312.586584 | 745.876776 | 737.332125 | 328.523493 |
| Average | 4.363531 | 281.300356 | 308.763337 | 676.790181 | 659.433718 | 300.803152 |

**Table 7**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.3125.

| Codebook size: **32** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 5.262231 | 306.532039 | 318.314149 | 761.346655 | 710.810851 | 343.742084 |
| PEPPER | 6.241595 | 338.618858 | 272.908139 | 571.875346 | 594.783631 | 347.997686 |
| BABOON | 5.171656 | 272.346143 | 289.707195 | 726.638634 | 723.566566 | 319.597288 |
| GOLDHILL | 4.481844 | 277.087057 | 313.025201 | 779.098764 | 755.606849 | 279.303908 |
| BARB | 6.675448 | 281.853149 | 315.750745 | 896.897654 | 877.698960 | 281.087256 |
| Average | 5.5665548 | 295.2874492 | 301.9410858 | 747.1714106 | 732.4933714 | 314.3456444 |

**Table 8**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.3750.

| Codebook size: **64** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 4.958598 | 311.440476 | 320.119462 | 745.345347 | 711.452695 | 320.128532 |
| PEPPER | 5.768537 | 305.034406 | 305.952327 | 636.967533 | 633.629908 | 315.600282 |
| BABOON | 6.728556 | 314.850105 | 323.926381 | 775.232423 | 768.088085 | 395.513511 |
| GOLDHILL | 8.795603 | 382.040368 | 391.577189 | 968.356788 | 934.453184 | 394.381707 |
| BARB | 11.21273 | 308.216570 | 312.251758 | 874.778777 | 846.853926 | 395.330121 |
| Average | 7.4928048 | 324.316385 | 330.7654234 | 800.1361736 | 778.8955596 | 364.1908306 |

**Table 9**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.4375.

| Codebook size: **128** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 11.888536 | 522.284262 | 534.054431 | 889.324333 | 866.079748 | 515.178820 |
| PEPPER | 16.421532 | 507.584419 | 570.995410 | 957.734356 | 914.451402 | 527.300916 |
| BABOON | 19.623416 | 405.582369 | 465.000805 | 964.673393 | 920.112334 | 836.092615 |
| GOLDHILL | 15.216410 | 697.720161 | 718.694109 | 1180.36544 | 1122.441935 | 419.510528 |
| BARB | 27.346822 | 521.941565 | 531.267813 | 1164.09897 | 1145.650131 | 506.671338 |
| Average | 18.0993432 | 531.0225552 | 564.0025136 | 1031.239298 | 993.74711 | 492.1654005 |

**Table 10**
The average computation time of the test images by using the six different algorithms with the bit rate = 0. 5000.

| Codebook size: **256** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 20.913288 | 875.945256 | 894.622455 | 799.356456 | 789.138474 | 665.040551 |
| PEPPER | 18.116559 | 750.584419 | 750.584419 | 997.987876 | 972.207816 | 567.219080 |
| BABOON | 28.028078 | 594.620811 | 563.620811 | 1011.56545 | 1032.44629 | 567.649493 |
| GOLDHILL | 29.701560 | 924.444311 | 556.342111 | 843.534555 | 827.919726 | 974.396983 |
| BARB | 27.619608 | 683.999751 | 692.899838 | 840.246767 | 830.791279 | 591.144534 |
| Average | 24.8758186 | 765.9189096 | 691.6139268 | 898.5382208 | 890.500717 | 699.450287 |

**Table 11**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.5625.

| Codebook size: **512** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 56.316994 | 1156.908559 | 1196.316994 | 1790.444554 | 1716.80639 | 1342.511572 |
| PEPPER | 82.002117 | 1950.230780 | 1851.809192 | 1387.908655 | 1357.86469 | 1112.212354 |
| BABOON | 63.433322 | 2010.197209 | 2187.580293 | 2178.565466 | 2212.438038 | 2458.561205 |
| GOLDHILL | 77.850099 | 1291.175546 | 1332.753876 | 2116.445100 | 2126.344435 | 1960.104396 |
| BARB | 115.21079 | 1296.291040 | 1123.215648 | 1396.123567 | 1386.554184 | 1102.154651 |
| Average | 78.9626644 | 1540.960627 | 1538.335201 | 1773.897468 | 1760.001547 | 1595.108836 |

**Table 12**
The average computation time of the test images by using the six different algorithms with the bit rate = 0.6250.

| Codebook size: **1024** | | | | | | |
|---|---|---|---|---|---|---|
| Image | Average computation time (sec) | | | | | |
| | LBG | PSO-LBG | QPSO-LBG | HBMO-LBG [18] | FF-LBG [19] | BA-LBG |
| LENA | 145.725844 | 3422.799272 | 3518.889707 | 4290.667555 | 4229.809396 | 3511.563824 |
| PEPPER | 140.346789 | 2262.336878 | 2558.192815 | 2773.786877 | 2723.738130 | 1892.857151 |
| BABOON | 156.576716 | 3376.804469 | 3386.370076 | 3914.987866 | 3848.840251 | 3917.097539 |
| GOLDHILL | 181.405248 | 2594.207355 | 2624.493640 | 2854.564565 | 2842.180938 | 1485.623620 |
| BARB | 211.115436 | 2847.841612 | 2826.772350 | 2254.343435 | 2221.664446 | 2131.103154 |
| Average | 167.0340066 | 2900.797917 | 2982.943718 | 3217.67006 | 3173.246632 | 2587.649058 |

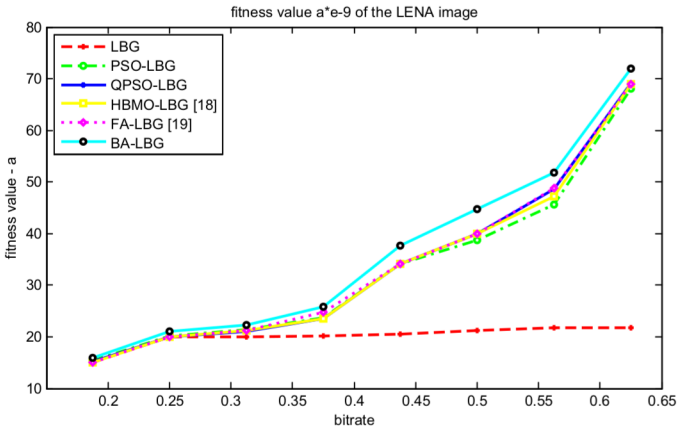



**Fig. 10.** The average fitness values of six vector quantization methods for LENA image.

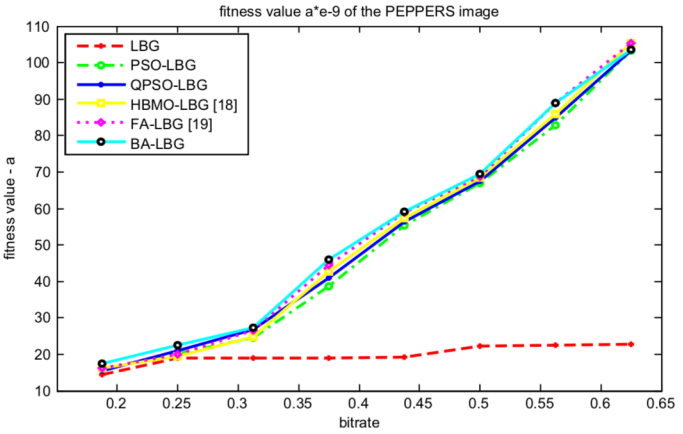



**Fig. 12.** The average fitness values of six vector quantization methods for PEPPERS image.

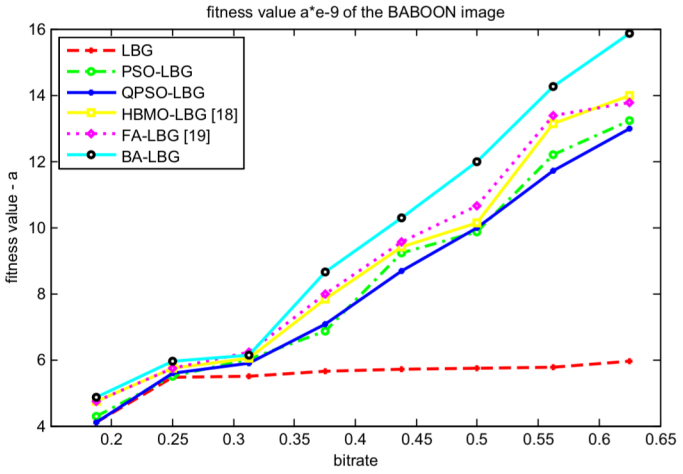



**Fig. 11.** The average fitness values of six vector quantization methods for BABOON image.

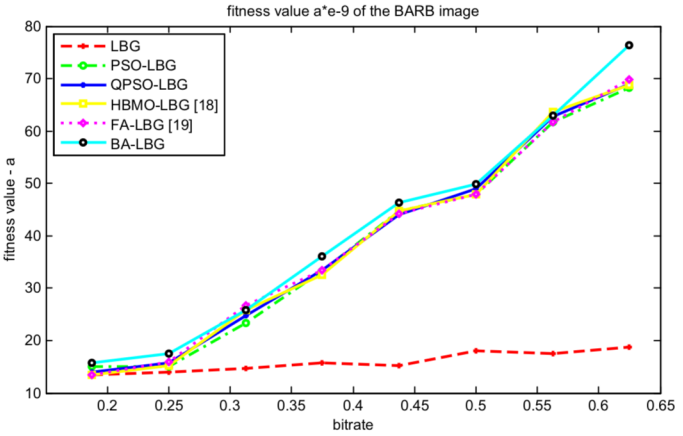



**Fig. 13.** The average fitness values of six vector quantization methods for BARB image.

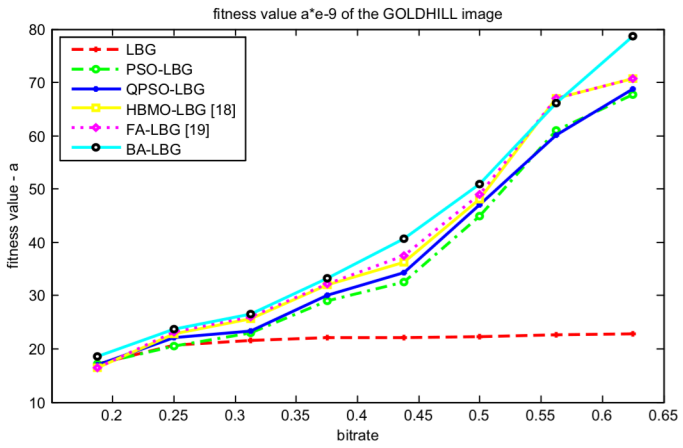



**Fig. 14.** The average fitness values of six vector quantization methods for GOLDHILL image.

## 5. Conclusions

In this paper, a Bat algorithm based vector quantization has been proposed for image compression. The Peak signal to noise ratio of vector quantization is optimized by employing BA technique. The algorithm has been investigated by varying all possible parameters of Bat algorithm for efficient codebook design and efficient vector quantization of training vectors. Intensification and diversification of the algorithm are achieved with Frequency-tuning and loudness parameter respectively. It is observed that the Bat algorithm peak signal to noise ratio and quality of the reconstructed image is superior to LBG, PSO-LBG, QPSO-LBG, HBMO-LBG and FA-LBG. From the simulation results, it is observed that BA-LBG is around 1.841 times faster convergence rate than that of the HBMO-LBG and FA-LBG. However, The BA-LBG algorithm requires some additional parameters compared with that of the PSO-LBG, QPSO-LBG and FA-LBG.

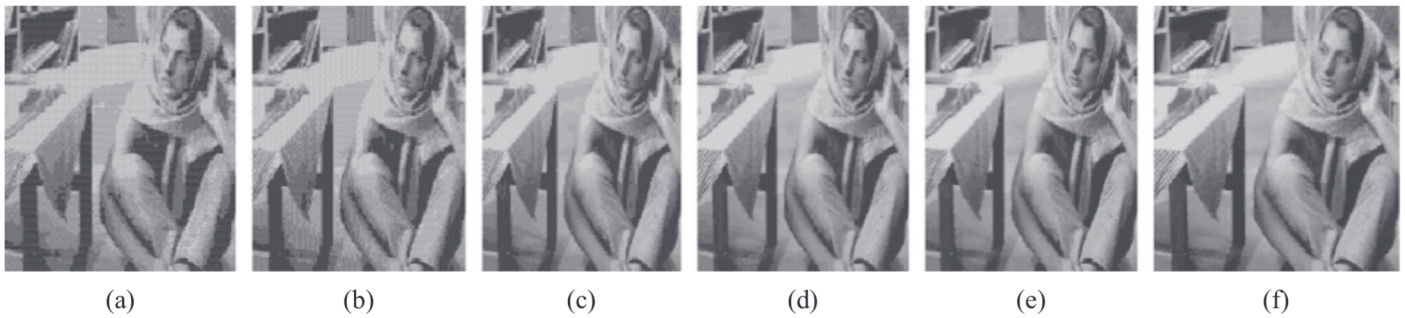



**Fig. 15.** Decompressed BARB image of six VQ techniques with codebook size of 256 (a) LBG (b) PSO-LBG, (c) QPSO-LBG (d) HBMO-LBG (e) FA-LBG (f) BA-LBG.

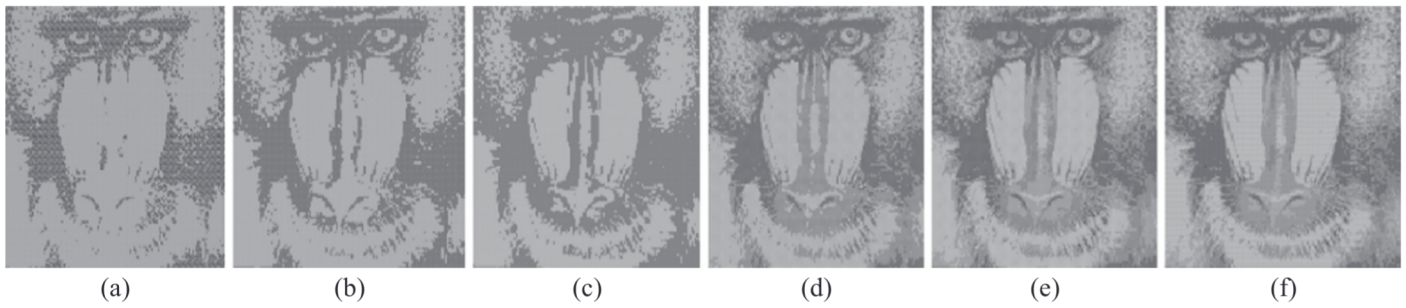



**Fig. 16.** Decompressed BABOON image of six VQ techniques with codebook size of 256 (a) LBG (b) PSO-LBG, (c) QPSO-LBG (d) HBMO-LBG (e) FA-LBG (f) BA-LBG.

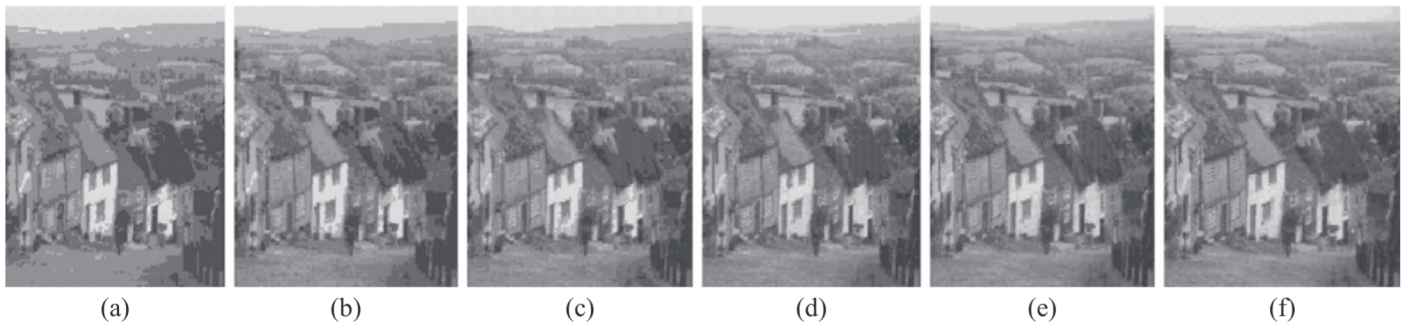



**Fig. 17.** Decompressed GOLDHILL image of six VQ techniques with codebook size of 256 (a) LBG (b) PSO-LBG, (c) QPSO-LBG (d) HBMO-LBG (e) FA-LBG (f) BA-LBG.

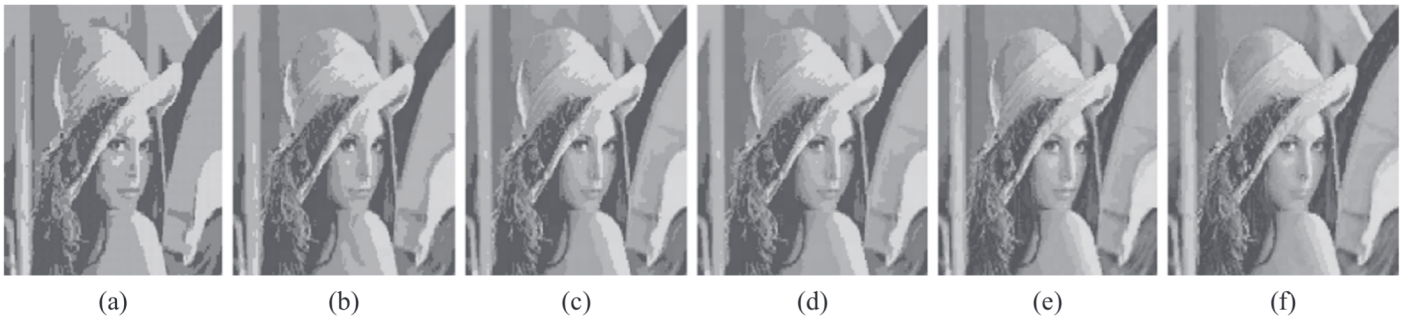



**Fig. 18.** Decompressed LENA image of six VQ techniques with codebook size of 256 (a) LBG (b) PSO-LBG, (c) QPSO-LBG (d) HBMO-LBG (e) FA-LBG (f) BA-LBG.

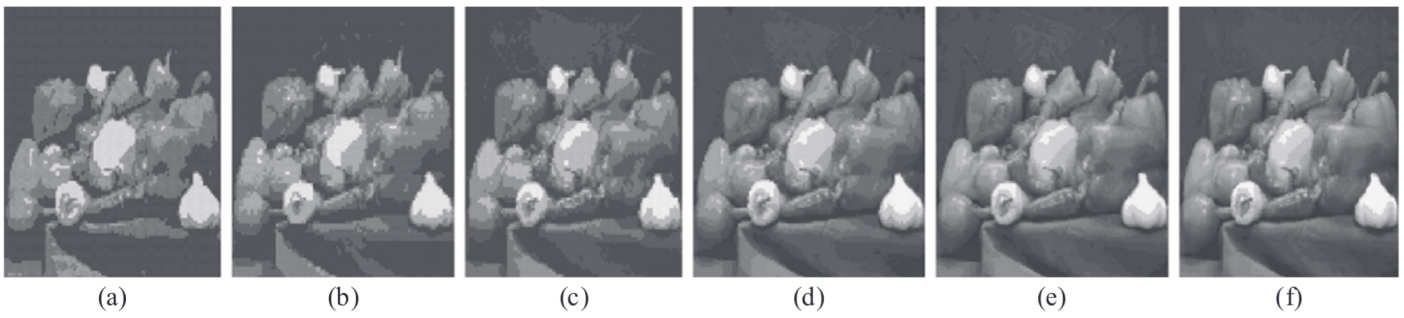
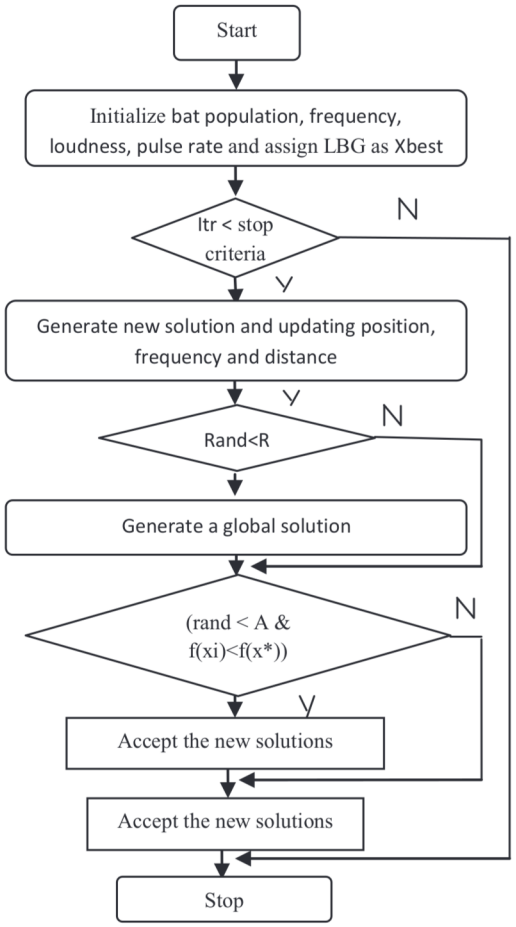



**Fig. 19.** Decompressed PEPPER image of six VQ techniques with codebook size of 256 (a) LBG (b) PSO-LBG, (c) QPSO-LBG (d) HBMO-LBG (e) FA-LBG (f) BA-LBG.

## Appendix: Flow chart of Bat Algorithm




## References

[1] R.M. Gray, Vector quantization, IEEE Signal Process. Mag. 1 (2) (1984) 4–29.
[2] D. Ailing, C. Guo, An adaptive vector quantization approach for image segmentation based on SOM network, Neurocomputing 149 (2015) 48–58.
[3] H.B. Kekre, Speaker recognition using vector quantization by MFCC and KMCG clustering algorithm, in: IEEE International Conferences on Communication, Information & Computing Technology (ICCICT), IEEE, Mumbai, 2012, pp. 1–5.
[4] C.W. Tsai, C.Y. Lee, M.C. Chiang, C.S. Yang, A fast VQ codebook generation algorithm via pattern reduction, Pattern Recognit. Lett. 30 (2009) 653–660.
[5] S.K. Frank, R.E. Aaron, J. Hwang, R.R. Lawrence, Report: a vector quantization approach to speaker recognition, AT&T Tech. J. 66 (2) (2014) 14–16.
[6] C.H. Chan, M.A. Tahir, J. Kittler, M. Pietikäinen, Multiscale local phase quantization for robust component-based face recognition using kernel fusion of multiple descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2013) 1164–1177.
[7] G.E. Tsekouras, D. Darzentas, I. Drakoulaki, A.D. Niros, Fast fuzzy vector quantization, in: IEEE International Conference on Fuzzy Systems (FUZZ), IEEE, Barcelona, 2010, pp. 1–8.
[8] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Trans. Commun. 28 (1) (1980) 702–710.
[9] G.E. Tsekouras, D.M. Tsolakis, Fuzzy clustering-based vector quantization for image compression, in: A. Chatterjee, P. Siarry (Eds.), Computational Intelligence in Image Processing, Springer Berlin, Heidelberg, 2012, pp. 93–105.
[10] M.R. Anderberg, Cluster Analysis for Applications, Academic Press, New York, 1973, pp. 162–163.
[11] G. Patane, M. Russo, The enhanced LBG algorithm, Neural Netw. 14 (2002) 1219–1237.
[12] A. Rajpoot, A. Hussain, K. Saleem, Q. Qureshi, A novel image coding algorithm using ant colony system vector quantization, in: International Workshop on Systems, Signals and Image Processing, Poznan, Poland, 2004, pp. 13–15.
[13] C.W. Tsaia, S.P. Tsengb, C.S. Yangc, M.C. Chiangb, PREACO: a fast ant colony optimization for codebook generation, Appl. Soft Comput. 13 (2013) 3008–3020.
[14] M. Kumar, R. Kapoor, T. Goel, Vector quantization based on self-adaptive particle swarm optimization, Int. J. Nonlinear Sci. 9 (3) (2010) 311–319.
[15] H.M. Feng, C.Y. Chen, F. Ye, Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression, Expert Syst. Appl. 32 (2007) 213–222.
[16] Y. Wang, X.Y. Feng, X.Y. Huang, D.B. Pu, W.G. Zhou, Y.C. Liang, et al., A novel quantum swarm evolutionary algorithm and its applications, Neurocomputing 70 (2007) 633–640.
[17] M.H. Horng, T.W. Jiang, Image vector quantization algorithm via honey bee mating optimization, Expert Syst. Appl. 38 (3) (2011) 1382–1392.
[18] M.H. Horng, Vector quantization using the firefly algorithm for image compression, Expert Syst. Appl. 39 (1) (2012) 1078–1091.

[19] C.C. Chang, Y.C. Li, J.B. Yeh, Fast codebook search algorithms based on tree-structured vector quantization, Pattern Recognit. Lett. 27 (10) (2006) 1077–1086.
[20] Y.C. Hu, B.H. Su, T.C. Chiang, Fast VQ codebook search for gray scale image coding, Image Vis. Comput. 26 (5) (2008) 657–666.
[21] H.B. Kekre, T.K. Sarode, S.R. Sange, S. Natu, P. Natu, Halftone image data compression using Kekre's fast code book generation (KFCG) algorithm for vector quantization, in: Technology Systems and Management, vol. 145, Communications in Computer and Information Science, 2011, pp. 34–42.
[22] H.B. Kekre, T.K. Sarode, S.D. Thepade, V. Vaishali, Kekre's fast codebook generation in VQ with various color spaces for colorization of grayscale images, in: Thinkquest, Springer Link, 2010, pp. 102–108.
[23] S. Yang, R. Wu, M. Wang, L. Jiao, Evolutionary clustering based vector quantization and SPIHT coding for image compression, Pattern Recognit. Lett. 31 (2010) 1773–1780.
[24] X. Li, J. Ren, C. Zhao, T. Qiao, S. Marshall, Novel multivariate vector quantization for effective compression of hyperspectral imagery, Opt. Commun. 332 (2014) 192–200.
[25] S.M. Hosseini, A. Naghsh-Nilchi, Medical ultrasound image compression using contextual vector quantization, Comput. Biol. Med. 42 (2012) 743–750.
[26] B. Huanga, Y. Wanga, J. Chen, ECG compression using the context modeling arithmetic coding with dynamic learning vector–scalar quantization, Biomed. Signal Process. Control 8 (2013) 59–65.
[27] P.K. Tripathy, R.K. Dash, C.R. Tripathy, A dynamic programming approach for layout optimization of interconnection networks, Eng. Sci. Technol. 18 (2015) 374–384.
[28] D. Tsolakis, G.E. Tsekouras, J. Tsimikas, Fuzzy vector quantization for image compression based on competitive agglomeration and a novel codeword migration strategy, Eng. Appl. Artif. Intell. 25 (2012) 1212–1225.
[29] G.E. Tsekouras, M. Antonios, C. Anagnostopoulos, D. Gavalas, D. Economou, Improved batch fuzzy learning vector quantization for image compression, Inf. Sci. (Ny) 178 (2008) 3895–3907.
[30] G.E. Tsekouras, A fuzzy vector quantization approach to image compression, Appl. Math. Comput. 167 (1) (2005) 539–5605.
[31] D. Tsolakis, G.E. Tsekouras, A.D. Niros, A. Rigos, On the systematic development of fast fuzzy vector quantization for gray scale image compression, Neural Netw. 36 (2012) 83–96.
[32] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), vol. 284, Studies in Computational Intelligence, Springer Berlin, 2010, pp. 65–74.
[33] A.H. Gandomi, X.S. Yang, S. Talatahari, S. Deb, Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization, Comput. Math. Appl. 63 (1) (2012) 191–200.
[34] G. Komarasamy, A. Wahi, An optimized K-means clustering technique sing bat algorithm, Eur. J. Sci. Res. 84 (2) (2012) 263–273.
[35] Z.Y. Du, B. Liu, Image matching using a Bat algorithm with mutation, Appl. Mech. Mater. 203 (1) (2012) 88–93.
[36] T.A. Lemma, F. Bin Mohd Hashim, Use of Fuzzy systems and Bat algorithm for energy modelling in a gas turbine generator, in: IEEE Colloquium on Humanities, Science and Engineering (CHUSER-2011), 2011, pp. 305–310.
[37] J.H. Lin, C.W. Chou, C.H. Yang, H.L. Tsai, A chaotic levy flight Bat algorithm for parameter estimation in nonlinear dynamic biological systems, J. Comput. Inf. Technol. 2 (2) (2012) 56–63.
[38] J. Kennedy, R.C. Eberhart, A new optimizer using particle swarm theory, in: Proceedings of Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[39] Q. Chen, J.G. Yang, J. Gou, Image compression method using improved PSO vector quantization, in: First International Conference on Neural Computation (ICNC 2005), vol. 3612, Lecture Notes on Computer Science, 2005, pp. 490–495.
[40] A. Baykasoglu, L. Ozbakor, P. Tapkan, Artificial bee colony algorithm and its application to generalized assignment problem, in: F.T.S. Chan, M.K. Tiwari (Eds.), Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, 2007, pp. 113–144.
[41] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 39 (2007) 459–471.
[42] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[43] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.
[44] D.P. Rini, S.M. Shamsuddin, S.S. Yuhaniz, Particle swarm optimization: technique, system and challenges, Int. J. Comput. Appl. 14 (1) (2011) 19–27.
[45] G.T. Chandra Sekhar, R.K. Sahu, A.K. Baliarsingh, S. Panda, Load frequency control of power system under deregulated environment using optimal firefly algorithm, Int. J. Electr. Power Energy Syst. 74 (2016) 195–211.
[46] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[47] <http://stackoverflow.com/questions/20513071/performance-tradeoff-when-is-matlab-better-slower-than-c-c>.