Image Anal Stereol 2018;37:35-54 doi: 10.5566/ias.1611
Original Research Paper

SAR IMAGE COMPRESSION USING ADAPTIVE DIFFERENTIAL EVOLUTION AND PATTERN SEARCH BASED K-MEANS VECTOR QUANTIZATION

KARRI CHIRANJEEVI✉,[1], UMARANJAN JENA[2]

[1]Department of Electronics and Communication Engineering, GMR Institute of Technology, Rajam, Srikakulam, Andhrapradesh, India; [2]Department of Electronics and Telecommunication Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla-768018, Odisha, India.
e-mail: chiranjeevi.k@gmrit.org, urjena@rdiffmail.com[2]
(Received August 26, 2016; revised August 18, 2017; accepted August 21, 2017)

ABSTRACT

A novel Vector Quantization (VQ) technique for encoding the Bi-orthogonal wavelet decomposed image using hybrid Adaptive Differential Evolution (ADE) and a Pattern Search optimization algorithm (hADE-PS) is proposed. ADE is a modified version of Differential Evolution (DE) in which mutation operation is made adaptive based on the ascending/descending objective function or fitness value and tested on twelve numerical benchmark functions and the results are compared and proved better than Genetic Algorithm (GA), ordinary DE and FA. ADE is a global optimizer which explore the global search space and PS is local optimizer which exploit a local search space, so ADE is hybridized with PS. In the proposed VQ, in a codebook of codewords, 62.5% of codewords are assigned and optimized for the approximation coefficients and the remaining 37.5% are equally assigned to horizontal, vertical and diagonal coefficients. The superiority of proposed hybrid Adaptive Differential Evolution and Pattern Search (hADE-PS) optimized vector quantization over DE is demonstrated. The proposed technique is compared with DE based VQ and ADE based quantization and with standard LBG algorithm. Results show higher Peak Signal-to-Noise Ratio (PSNR) and Structural Similiraty Index Measure (SSIM) indicating better reconstruction.

Keywords: differential evolution (ADE); image compression; Linde-Buzo-Gray (LBG); Pattern Search (PS); vector quantization.

INTRODUCTION

Synthetic Aperture Radar (SAR) images are high resolution images that carries the amplitude and phase information of object/target captured and processes with transferable radar. These are beneficial for military applications, plant investigations, remote sensing and Earth scientists. Transmitting or storing of SAR images requires large time and high data storage devices, consequently necessitates the use of efficient and effective image compression techniques. SAR image has less spatial correlation, identifying homogeneous regions and high dynamic range (Zeng and Cumming, 2001) is difficult as compared to ordinary optical images. So the compression techniques to solve an ordinary optical image are not suitable for compression of SAR images. SAR Image compression plays crucial role in the field of multimedia applications, streaming data on the Internet, data storage, wire and wireless communication, chip designing and computer to computer communication. Image Compression is a technique of representating the image that results in reduction of bits per pixel and processing time. Image compression is performed in three processing stages: pixel transforms, Quantization and entropy coding. Pixel transformation is a powerful process for image compression where spatial domain image is transformed into a frequency domain image with some transformation techniques like DWT (JPEG-2000 and JPEG-XR), DCT (JPEG,). In this work the focus is on Quantization. It has two variants, one is a scalar, applicable for one dimensional (speech, voice) and another is vector quantization/block quantization/pattern matching applicable for two dimensions (image). According Shannon's rate-distortion information theory, compression with vector quantization performance is better to scalar quantization (Liu and Ling, 2015).Vector quantization is a process of representing high two dimensional vectors/image into a low dimensional vector/image (called codebook) based on the minimum Euclidean distance/ average distortion. The elements in the codebook are called

codewords. The minimization of the average distortion is a nonlinear problem, which is generally achieved by a gradient-descent-based iterative procedure called the generalized Lloyd algorithm (GLA)/LBG Algorithm (Linde *et al*., 1980). The goal of VQ is to find a codebook which minimizes the Euclidean distance between training vector and codewords. LBG algorithm is simple, adaptable and flexible, but it suffers with local optimal problem. The LBG algorithm does not guarantee the global best solution since its solution depends on the initial solution of the algorithm. So Patane *et al.* (2002) proposed an enhanced LBG algorithm that avoids the local optimal problem. In genetic algorithm, crossover and mutation plays vital role. Krishna *et al.* (1997) obtained optimal fast codebook with genetic algorithm by applying genetic algorithm on k-means clustering with the help of Gradient descent in which k-means algorithm is used as crossover operation and mutation operation is distance based. Chen (2012) hybridized the Genetic algorithm and LBG algorithm for optimizing the center of clusters. Zheng *et al*. (1997) applied genetic algorithm to speed up the LBG algorithm by using an appropriate fitness function. The results obtained are compared with those of non-genetic algorithm and proved effective but time consuming.

George and Dimitrios (2012) incorporated the c-means and the fuzzy c-means in a uniform fashion for balancing the speed and efficiency of vector quantization. Zhao and Liu (2013) presented a clonal cluster method based on the manifold distance, which produces a final codebook with the help of optimization technique. Horng and Jiang (2011) proposed an Artificial Bee Colony (ABC) based VQ that shows improvement in PSNR with good reconstructed image quality as compared to LBG, PSO, QPSO and HBMO. Rajpoot *et al*. (2004) designed a codebook by vector quantizing the DWT transformed wavelet coefficients with the help of Ant Colony Optimization (ACO) algorithm. They designed a codebook by identification of the edges of the graph and arranging the wavelet coefficients in a bidirectional graph. It was found that quantization of zero-tree vectors using ACO outperforms LBG algorithm, but ACO convergence time is high. Tsaia *et al*. (2013) observed that during the convergence process of ACO for Codebook Generation Problem (CGP), patterns or sub-solutions reach their final states at different times. Also, most of the patterns are assigned to the same codewords after a certain number of iterations. Particle Swarm Optimization (PSO) vector quantization, based on updating the global best (*gbest*) and particle best (*pbest*) solution (Chen *et al*., 2005) outperforms LBG algorithm.The gbest holds highest fitness value among all populations and *pbest* holds the best fitness value of corresponding particle. The Feng *et al*. (2007) suggested Evolutionary fuzzy particle swarm optimization algorithm which is a combination of PSO and adaptive Fuzzy Inference Method (FIM) to obtain better global performances than LBG learning algorithms.

Quantum Particle Swarm Algorithm (QPSO) was proposed by Wang *et al*. (2007) to solve the 0-1 knapsack problem. The QPSO performance is better than PSO; it computes the local points from the pbest and gbest for each particle and updates the position of the particle by choosing appropriate parameters *u,* a random number that lies between 0 and 1 and *z* which is non-negative constant and is less than 2.8. Poggi *et al*. (2001) proposed Tree-structured product-codebook vector quantization, which reduces encoding complexity even for large vectors by combining the tree-structured component codebooks and a low-complexity greedy procedure. Hu *et al*. (2008) proposed a fast codebook search algorithm based on triangular inequality estimation. Sanyal *et al*. (2013) applied a new approach for the selection of chemotaxis steps of basic Bacterial Foraging Optimization Algorithm (BFOA) which leads to the development of a near optimal codebook for image compression with good reconstructed image quality and high peak signal to noise ratio. Fuzzy membership function is optimized by the modified Bacterial Foraging Optimization and compared the results with other optimization techniques. Horng *et al*. (2011) applied honey bee mating optimization algorithm for Vector quantization. HBMO has high quality reconstructed image and better codebook with small distortion compared to PSO-LBG, QPSO-LBG and LBG algorithm.

Horng (2012) applied a Firefly Algorithm (FA) to design a codebook for vector quantization. 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 practice. Firefly algorithm is encouraged by social activities of fireflies and the occurrence of bioluminescent communication. Fireflies with lighter intensity values move towards the brighter intensity fireflies and if there is no brighter firefly then it moves randomly. Object-based VQ was proposed by Abouali (2015) based on an iterative process of LBG algorithm, max-min algorithm and multi-object applications. The proposed method takes the advantage of well suitable high dimensional problem technique which is an adaptive differential evolution (ADE) and near around/local search technique which is pattern search (PS) for

effective codebook design by taking the initial solution of K-Means clustering algorithm/LBG Algorithm as one of the population/solution. In this work the modified version of differential evolution and pattern search are cascaded to form a Hybrid ADEPS (hADE-PS) algorithm. The SAR image to be vector quantized is transformed to wavelet domain with the help Biorthogonal Discrete Wavelet Transform (DWT) because most of the energy is concentrated on the low frequency band. The transformed SAR image is now vector quantized by using K-Means clustering algorithm on which hADE-PS works for efficient codebook design. Adaptive differential evolution is developed and tested on benchmark functions and results obtained are verified and found to be better as compared to ordinary differential evolution in which mutation operation is random. Whereas ADE follows a specific order/strategy while selecting population for mutation operation. To improve compression ratio with considerable PSNR and reconstructed image quality, a perfect codebook design is crucial for the researcher. So in this paper, a hybrid adaptive differential evolution and pattern search is proposed for efficient codebook design by optimizing the solution of K-Means clustering algorithm. The optimized codebook and corresponding index table values are further coded with a run length coding followed by a Huffman coding at the transmitter section and reverse operation at the receiver section. The proposed method is compared with the Differential Evolution (DE) based VQ and Adaptive Differential Evolution (ADE) based VQ and experimentally proved that it has superior Peak Signal Noise Ratio (PSNR), Mean Square Error (MSE), fitness function, bits per pixel, compression ratio and Structural Similiraty Index Measure (SSIM). This paper is organized in five sections including the introduction. In section 2, proposed framework for SAR Image compression and recent methods of VQ is discussed along with their algorithms. The proposed method of hybrid adaptive differential evolution and pattern search VQ algorithm is presented with the procedure in section 3. The results and discussions are given in section 4. Finally the conclusion is given in section 5.

MATERIALS AND METHODS

CONTRIBUTION

In the proposed method of SAR image compression, the differential evolution is modified as adaptive differential evolution for efficient codebook design and for further improvement a pattern search algorithm is applied on the solution of adaptive differential evolution. The encoding and decoding procedure of proposed SAR image compression is shown in Fig. 1. At the transmitter/encoding section, the SAR image to be vector quantized for compression is transformed to wavelet domain by using Biorthogonal Discrete Wavelet Transform and further partitioned into non overlapping blocks called input vector. The input vectors are clustered based on the minimum Euclidean distance between the input vector and codewords of codebook by means of iterative method called K-means clustering algorithm/LBG algorithm. The obtained codebook is optimized by the proposed hybrid adaptive differential evolution and pattern search algorithm. The optimized codebook of index table contains repeated information, so it is coded by variable Run-length coding followed by Huffman coding. At the receiver /decoder section, the index values are retrieved by Huffman decoding and Run-length decoding and from these index values the corresponding codewords are obtained from the receiver end codebook and rearranged to get a reconstructed/decompressed image.

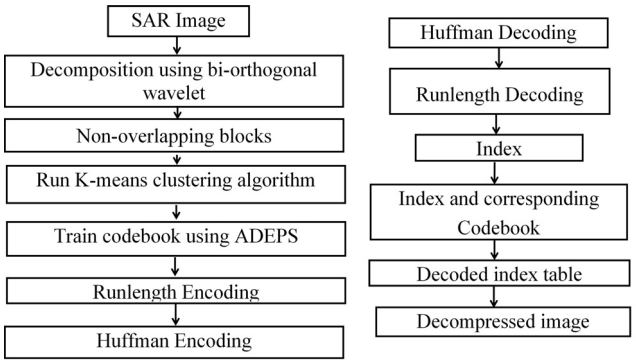

Fig. 1. *Encoding and decoding process of proposed vector quantization.*

VECTOR QUANTIZATION

The vector quantization is a one of the block coding technique for image compression. Codebook design is an important task in the design of VQ that minimize the distortion between reconstructed image and original image with less computational time. Fig. 2 shows the encoding and decoding process of vector quantization. The image (size $N \times N$) to be vector quantized is subdivided into N_b $(N/n \times N/n)$blocks with size $n \times n$ pixels. These divided image blocks or training vectors of size $n \times n$ pixels are represented with X_i $(i = 1, 2, 3, \ldots, N_b)$. The Codebook has a set of codewords, C_i (where $i = 1 \ldots N_c$) is the i^{th} codeword. The total number of codewords in Codebook is N_c. Each subdivided image vector is approximated by the index of codewords based on the minimum Euclidean distance between corresponding vector and code

words. The encoded results are called an 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/fitness function (*D)* between training vectors and the codebook is given as

$$D = \frac{1}{N_c}\sum_{j=1}^{N_c}\sum_{i=1}^{N_b} u_{ij} \cdot \left\| X_i - C_j \right\|^2 . \quad (1)$$

Subject to the following constraints:

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

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,....,*N_c must satisfy $R_j \supset \left\{ x \varepsilon X : d(\mathrm{c}, \mathrm{C}_j) < d(\mathrm{x}, \mathrm{C}_k), \forall\, \mathrm{k} \neq \mathrm{j} \right\}$. (3)

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

$$C_j = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i \qquad x_i \in R_j , \quad (4)$$

where N_j is the total number of vectors belonging to R_j

DECOMPOSITION USING BIORTHOGONAL WAVELET

In 1970's images are decomposed with Discrete Cosine Transform (DCT) in which most of the energy is concentrated in DC coefficients, that helps for high compression with considerable artifact effect. The image compression has leaped to a new level (Daubechies, 1988) with the introduction of Discrete Wavelet Transform (DWT). Unlike DCT, the DWT provides both spatial and frequency information about the image. The DWT decomposes the image into four coefficients; approximation (low-low frequency), horizontal (low-high frequency), vertical (high-low frequency) and diagonal (high-high frequency) coefficients. These coefficients are obtained with the parallel combination of low pass filter and high pass filter and down samplers as shown in Fig. 3. Fig. 4 shows the three dimensional view of approximation, horizontal, vertical and diagonal coefficients of a Flight Line SAR image. It is observed that approximation coefficients carry much information about the input image as compared to other coefficients whereas all horizontal, vertical and diagonal coefficients spread their values near around to particular values, that helps for good clustering results in better image compression as shown in Fig. 4d. For the sake of fidelity of reconstructed image quality one level decomposition is applied, the same can be applied to more than one decomposition levels for a high degree of compression at the cost of time. Like in JPEG-2000, the wavelet used in our work for decomposition is biorthogonal wavelet because of its simple design and option to build symmetric wavelet functions. In the proposed method, optimization technique spent much time in the codebook design of approximation coefficients and less time for remaining, because the reconstructed image quality depends predominantly on approximation coefficients.

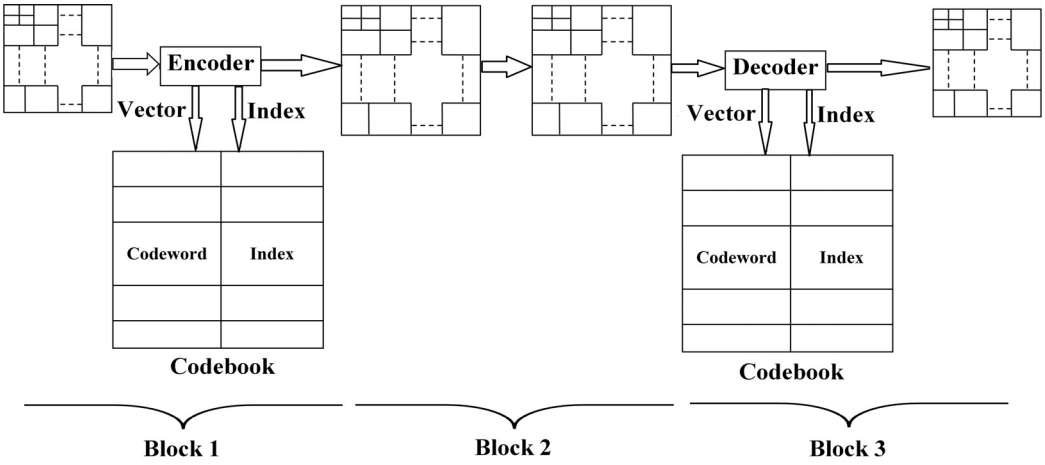

Fig. 2. *Encoding and decoding process of vector quantization.*

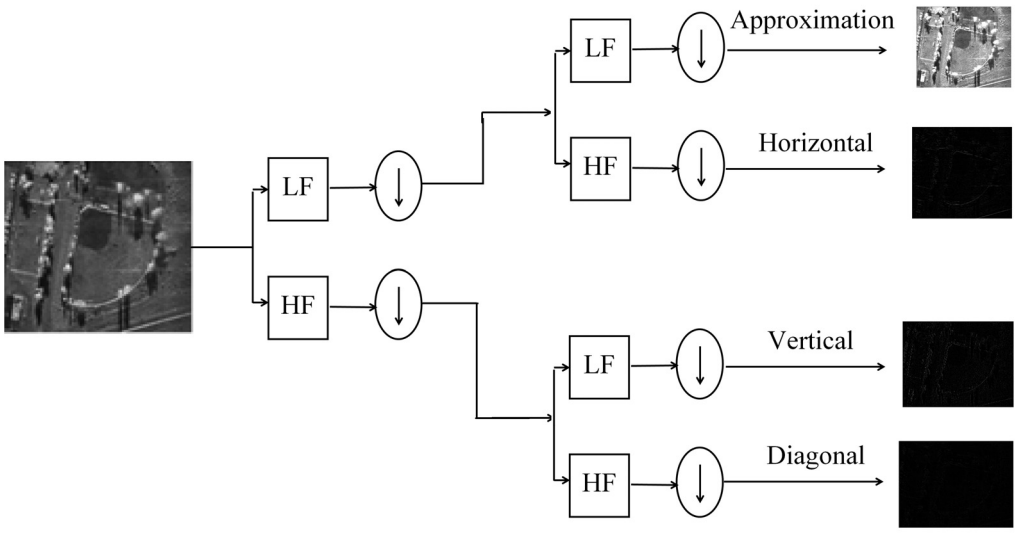

Fig. 3. *Wavelet decomposition.*

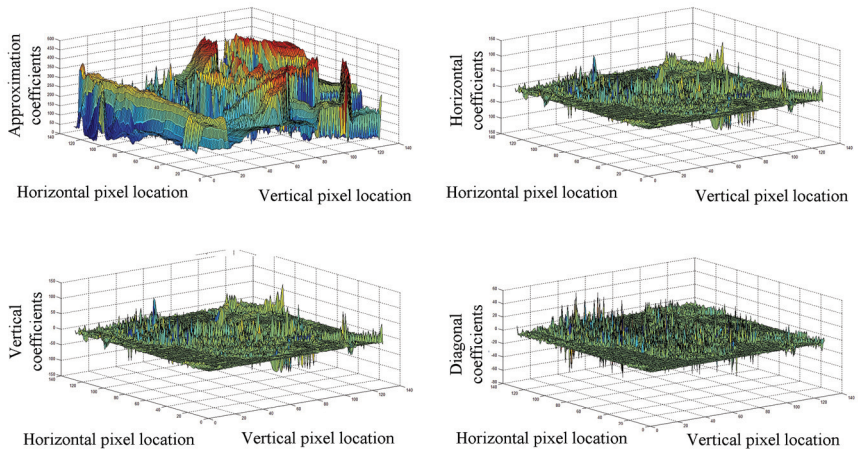

Fig. 4. *Three dimensional view of Approximation, Horizontal, Vertical and Diagonal coefficients of a Flight Line SAR image.*

In a codebook of eight codewords, using the proposed VQ, five codewords are assigned and optimized for the approximation coefficients, one for horizontal, one for vertical and one for diagonal coefficients. Similarly, in a codebook size of 16, 32, 64, 128, 256 and 512, approximation coefficients takes codewords of 10, 20, 40, 80, 160 and 320 respectively and remaining codewords 6, 12, 24, 48, 96, 192 are assigned for the other three coefficients with each containing a codeword of 2, 4, 8, 16, 32, 64 respectively. While optimizing the codebook population/ solution, maximum and minimum values are selected based on the max and min value of corresponding coefficients. After successful codebook optimization with the proposed hADE-PS, index table and codebook are coded by Run Length Coding (RLE) and Huffman coding. Runlength coding is a lossless coding, aiming to reduce the amount of data needed for storage and transmission. It represents consecutive runs of the same value in the data as the value, followed by the count or vice versa. If all the values in the original data are same, RLE can reduce data to just two values otherwise double of the original data. Therefore, RLE should only be used in cases where runs of the same value are expected. Huffman coding is a lossless variable length coding, best fit for compressing a data/ image obtained from run length coding. It uses fewer bits to represent frequent symbols and more bits to represent infrequent symbols. The performance of Huffman coding purely depends on the effective development of Huffman tree with minimum weighted path length. The time complexity of Huffman coding is $O\ (Nlog_2\ N)$ where each iteration $O\ (log_2 N)$ time to determine the cheapest weight and there would be $O\ (N)$ iterations.

GENERALIZED LBG VECTOR QUANTIZATION ALGORITHM

The most commonly used methods in VQ are the Generalized Lloyd Algorithm (GLA) which is also called Linde-Buzo-Gary (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 $D_1 = \infty$.

Step 2: 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 step 2–4.

HYBRID ADAPTIVE DIFFERENTIAL EVOLUTION AND PATTERN SEARCH (HADE-PS) VQ ALGORITHM

To examine the extremely multimodal space, a two phase hybrid method recognized as hybrid Differential Evolution and Pattern Search (hADE-PS) is employed. In this algorithm, DE is used for global exploration and the pattern search is employed for local search. The first phase is explorative, employing a classical ADE to identify hopeful areas of the search space. The best solution initiated by ADE is then polished using the PS method during a consequent exploitative phase. In order to find the advantage of projected hADE-PS approach, the results are compared with the DE algorithm. In the subsequent section, proposed ADE is employed for for efficient codebook design.

DIFFERENTIAL EVOLUTION VECTOR QUANTIZATION

The LBG algorithm distortion becomes smaller after recursive execution. Actually, the LBG algorithm can guarantee that the distortion will not increase from one iteration to the next iteration. However, the resulting codebook may not be the optimum one and the initial condition will significantly influence the results (Chiranjeevi and Umaranjan, 2015). Therefore, in the LBG algorithm more attention should be given to the choice of the initial codebook. The problems with the LBG algorithm can be overcome by using differential evolution. Differential evolution is meta-heuristics algorithm introduced for large dimensional problems in the year 1995 by (Stron and price, 1995). Differential evolution is applicable in the field of engineering and science problems because of its high speed of convergence, less expensive, easy to implement, negligible parameter tuning and real coding. The working process of DE is similar to the GA in which three steps are crossover, mutation and selection. The DE algorithm initializes the populations/solutions X ($X_1, X_2, X_3,$-------- X_N) of size N and D dimensions in between upper and lower limits of the problem (0,255), in this paper D is equal to 16 (block size = 16). The population values are initialized randomly within the limited upper (X_{upper}) and lower (X_{lower}) bounds as given in Eq. 5, here as the image is gray-scale image upper bound is 255 (white) and lower is 0 (black).

$$X_K = X_{lower} + rand(1,1) \times (X_{upper} - X_{lower}) \qquad K = 1,2,3,\text{----}N, \quad (5)$$

where rand (1,1) is a random number of size one lying between 0 and 1

Calculate the fitness/objective function of all populations ($f(X_1)$, $f(X_2)$, $f(X_3)$,-------- $f(X_N)$) and considered these as old generation. In each iteration the new generation of same population is generated with three steps: mutation, crossover and selection. In mutation operation, for all populations in each iteration a trial vector/donor vector V_K is created by adding the weighted difference of randomly selected two populations multiplied by the scalar multiple control parameter (F) to a random third population as given in Eq. 6, these three populations are selected randomly from the initialized populations.

$$V_k^i = X_{r1}^i + F \times (X_{r2}^i - X_{r1}^i), \quad (6)$$

where r_1, r_2 and r_3 are randomly selected numbers lying between 1 and N, i is the i^{th} iteration and F is scaling factor lying between 0 and 2. The next step of DE is crossover; to strengthen the potential diversity of solution and to generate a new solution called target vector. DE offers two kinds of crossovers named Exponential and Binomial, among two we did with Binominal in which for each trial vector an offspring vector U_K is created as in Eq. 7, if the generated random number is less than the control parameter crossover rate (CR). The crossover rate lies between 0 and 1. DE offers several variants or mutation strategies like DE/rand/1, DE/best/2 etc... (Price *et al*., 2005). The last stage of DE is the selection process for maintaining the constant population size in successive generations and it selects either target

vector or trial vector in the next generation based on their fitness value which follows the Darwinian principle given in Eq. 8.

$$U_k = \begin{cases} V_K, & if\ rand(0,1) < CR \\ X_K, & Else \end{cases}, \quad (7)$$

$$X_K(t+1) = \begin{cases} V_K(t), & if\ f(U_K(t)) \le f(X_K(t)) \\ X_K(t), & if\ f(U_K(t)) > f(X_K(t)) \end{cases}. \quad (8)$$

DE vector quantization algorithm:

Step 1: Initialize population size (N), scaling factor (F), crossover constant (CR), maximum number of iterations, tolerance, lower limit and upper limit.

Step 2: Evaluate the fitness of the population using Eq. 1 and generate a trial vector V_K with the help of Eq. 6.

Step 3: Crossover operation on population with crossover constant and generation of target vector U_K with the help of Eq. 7.

Step 4: Replace the old generation with newly generated population which are generated based on the fitness value of target vectors U_K or X_K.

Step 5: repeat step 2 to 4 until stopping criteria.

PATTERN SEARCH VECTOR QUANTIZATION

Hooke and Jeeves (1960) developed a Pattern search algorithm in the year 1960 for getting a solution of real time numerical and engineering problems which are unsolved by classical methods. Pattern search doesn't require the gradient of the problem to be optimized, so it is a derivative free or direct search optimization technique. Pattern search optimizes the objective function with two types of moves. First step is an exploratory move where, step of the movement of the solution is small in the direction of low/high objective function values and second is pattern move where, step of the movement of the solution is large in the direction of low/high objective function values.

Let starting point $X^{(0)}$, acceleration factor a, perturbation vector P_0 and perturbation tolerance vector T. Initialize the current perturbation vector: $P \leftarrow P_0$. Find new solution $X^{(1)}$ by exploratory search around $X^{(0)}$. If $X^{(1)}$ fitness value is not better than $X^{(0)}$ then, Reset all of the perturbations to ½, *i.e.*, $P \leftarrow P/2$. If any member of P is now smaller than its corresponding perturbation tolerance in T, then exit with $x^{(0)}$ as the solution. If $X^{(1)}$ fitness value is not better than $X^{(0)}$ then, reset the perturbation vector to its original value, *i.e.*, $P \leftarrow P_0$ and follow the Pattern Move. Pattern move generate $X^{(2)}$ from $X^{(0)}$ through $X^{(1)}$ which is given by $X^{(2)} = X^{(0)} + a \times (X^{(1)} - X^{(0)})$. Now apply exploratory search on $X^{(2)}$ and find the fitness $f(X^{(2)})$, if it is not better than $f(X^{(1)})$ then replace solution $X^{(0)}$ with solution $X^{(1)}$. If $f(X^{(2)})$ is better than or equal to $f(X^{(1)})$ then, replace solution $X^{(0)}$ with solution $X^{(1)}$ and $X^{(1)}$ with solution $X^{(2)}$. In this paper the initial solution $X^{(0)}$ is the outcome of the Adaptive differential evolution. At the initial iteration, perturbation vectors are initialized with the positions [0.5 0.5], [0.5 -0.5], [-0.5 0.5] and [-0.5 -0.5] with acceleration factor is equal to one. To generate a next solution $X^{(1)}$ those obtained from the perturbation vectors are added to $X^{(0)}$ i.e $X^{(0)}$ + [0.5 0.5], $X^{(0)}$ + [0.5 -0.5], $X^{(0)}$ + [-0.5 0.5] and $X^{(0)}$ + [-0.5 -0.5] as shown in Fig. 5. Calculate the fitness of $X^{(1)}$, if better than $X^{(0)}$ then perturbation vectors are consi-dered as successful perturbation vectors and algorithm replace the $X^{(0)}$ with $X^{(1)}$. Whenever successful per-turbation vectors occurred the algorithm will shift its state to pattern move where the acceleration factor is multiplied by a factor 2 so acceleration factor is also called as expansion factor. In the next iteration the new solution is $X^{(1)}$ +2×[0.5 0.5], $X^{(1)}$ +2×[0.5 -0.5], $X^{(1)}$ + 2×[-0.5 0.5] and $X^{(1)}$ +2×[-0.5 -0.5] this process is repeated until stopping criteria/maximum iteration. In process, if any perturbation vectors resultant fitness value not better than the initial/current fitness than perturbation vectors are called unsuccessful and the same solution carried to the next iteration. In this situation the algorithm acceleration factor is divided by a factor 2 so called contraction factor.

Pattern search vector quantization algorithm:

Step 1: Initialize number of iterations, dimensions of the problem, mesh contraction factor/mesh expansion factor (P), solutions (K).

Step 2: check the convergence for all possible solutions (X_k) where $k=1, 2, 3, \ldots\ldots K$.

Step 3: Calculate the objective function $f(X_k)$, and with the help of exploratory move calculates the step of search S_K.

Step 4: If new objective function $f(X_k+S_k)$ is less than $f(X_k)$, then new solution $X_{k+1} = X_k+S_k$ otherwise $X_{k+1} = X_k$.

Step 5: update mesh contraction factor/mesh expansion factor (P).

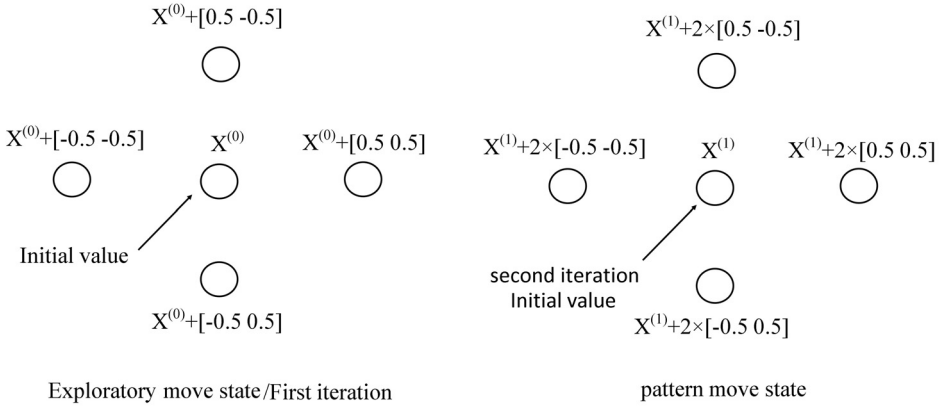

Fig. 5. *Exploratory move state and pattern move state of pattern search.*

ADAPTIVE DIFFERENTIAL EVOLUTION

The success of DE in solving a specific problem crucially depends on appropriately choosing donor vector generation strategies. In ordinary differential evolution, for a donor vector generation the selection of three populations is random as in Eq. 5, that results in a local optimal solution because the difference of two randomly selected populations nearly equals to largest solution and in addition to it the added population to the above difference results in a value that lies out of search space. As there is no particular pattern being followed in the selection of populations, it results in local optimal solution. So to overcome this we propose an adaptive differential evolution, to improve the performance, effectiveness and robustness of differential evolution for efficient vector quantization of image which leads to better image compression with good reconstructed image quality. The proposed novel differential evolution follows an intelligent selection of the population for donor vector V_K that reduce the distortion (D) between image to be compressed and codebook which is to be optimized. The ADE divides the population into two equal groups based on the descending fitness values. The first group is exploited for local search of the codebook, as the populations are arranged in descending order, the difference magnitude of two successive populations is less, which is added to the predecessor population. Hence the predecessor population explores for a solution around it as shown in Fig. 6. The same procedure is repeated for the first *N/2* populations among the available N descended populations.

For the global search of the codebook both the groups are harnessed. The donor vector generation is obtained from the difference of least fitness population (N^{th} element - second group) and the highest fitness population (1st element - first group) is added to $(N-1)^{th}$ population. Similarly

The difference of the next least fitness population ($(N-1)^{th}$ element - second group) and the precedent highest fitness population (2nd element - first group) is added to $(N-2)^{th}$ population and the same procedure is repeated for the remaining populations as depicted in the Fig. 6.

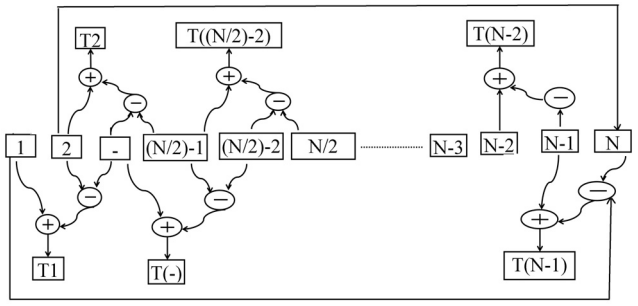

Fig. 6. *Diagrammatic representation of proposed ADE.*

The donor vector for N^{th} population is generated on the same lines as that of ordinary differential evolution. This complete process is repeated for various iterations until distortion between codebook and image to be compressed is a global minimum. To validate the performance of ADE, it is compared with DE based on nine benchmark functions. The simulation results show that ADE promises competitive performance not only in the PSNR, but also in the quality of reconstructed images. The algorithm of ADE is as similar as DE except the selection of three populations in a donor vector generation. The idea of using an adaptive selection of populations is to make ADE computationally more efficient in obtaining local and global solutions for the codebook.

Table 1. *Simulation results of Benchmark functions of GA, FA, DE and ADE.*

Dimension = 2, population = 50, iterations = 1000					
Function	Measure	GA	FA	DE	ADE
Akely	minimum	0.0263	1.2898	8.32E-07	3.64E-07
(-4, 5)	mean	0.0624	1.4871	6.17E-06	2.04E-06
Global min= 0	std.	0.0409	0.1476	9.18E-06	1.32E-06
Powell	minimum	0.0011	5.89E-06	3.34E-04	7.44E-05
(-4, 5)	mean	0.0091	0.0127	0.003	0.0015
Global min= 0	std.	0.0058	0.0315	0.0032	0.0014
Beale	minimum	0.0263	1.78E-07	1.01E-07	4.75E-08
(-4.5, 4.5)	mean	0.2321	1.14E-04	9.23E-06	2.37E-05
Global min= 0	std.	0.3739	1.48E-04	1.53E-05	2.71E-05
Bird	minimum	-106.765	-106.765	-106.765	-106.765
(-2pi, 2pi)	mean	-102.874	-106.753	-106.764	-106.764
Global min= 106.7	std.	8.2024	0.0152	6.39E-04	3.65E-04
Bukin4	minimum	1.30E-02	4.55E-05	4.70E-05	2.95E-05
(-15, 5)	mean	0.0035	0.0013	7.6657	2.2355
Global min= 0	std.	0.0086	7.1094	16.0819	2.8937
Chichinadze	minimum	-42.4972	-42.9441	-41.7684	-42.9444
(-30, 30)	mean	-42.4972	-42.9224	-41.8289	-42.8605
Global min=-43.31	std.	0.1414	0.0256	0.1365	0.1512
crosslegtable	minimum	-0.093	-0.0198	-0.013	-4.56E-04
(-10, 10)	mean	-0.0049	-0.0146	-0.6511	-4.69E-04
Global min= -1	std.	0.0035	5.997	5.3809	5.22E-05
goldsteinprice	minimum	3	3	3	3.00E+00
(-2, 2)	mean	3	3	3	3.00E+00
Global min = 3	std.	2.92E-15	4.15E-15	1.24E-06	4.93E-07
himmelblau	minimum	0.0023	2.43E-05	7.01E-07	4.87E-07
(-5, 5)	mean	0.312	0.0042	3.37E-04	8.28E-04
Global min = 0	std.	0.0015	0.0051	7.33E-04	0.0016
schweffel	minimum	-587.966	-637.909	-837.966	-837.966
(-500, 500)	mean	-555.056	-658.499	-837.966	-837.966
Global min = -837.9	std.	112.3693	101.39	7.71E-08	5.52E-08
testtubeholder	minimum	-10.8723	-10.8722	-10.8723	-10.8723
(-10, 10)	mean	-10.6449	-10.8713	-10.8718	-10.8723
Global min = -10.87	std.	0.203622	9.01E-04	0.0017	6.79E-05
zettl	minimum	-0.0038	-0.0038	-0.0038	-0.0038
(-5, 5)	mean	-0.0038	-0.0037	-0.0038	-0.0038
Global min = -0.0037	std.	2.89E-09	7.61E-05	5.44E-10	5.13E-09

PERFORMANCE EVALUATION OF ADE ALGORITHM

In this section some comparisons between the genetic algorithm, firefly algorithm, DE and ADE using twelve numerical benchmark functions is demonstrated. The benchmark functions chosen for validation of Adaptive ADE over other optimizations are Ackley, Powell, Beale, Bird, Bukin4, Chichinadze, Crosslegtable, Goldsteinprice, Himmelblau, Schweffel, Testtubeholder and Zettl (Yao *et al*., 1999). Initially the algorithm is validated by taking dimension = 2, population = 50, iterations = 1000 and performance measuring parameters like minimum, mean and standard deviation are considered. Table 1 shows the 12 benchmark functions and its corresponding range and theoretical minimum value. All the algorithms independently run for 50 times and minimum value is the minimum of objective function values of 50 independent runs. Mean or average is the ratio of the sum of the minimum value obtained to that of the number of independent runs. The mean value near to zero indicates better performance of the algorithm. The word '*std*' is the standard deviation, which is equal to the

square root of variance; a standard deviation close to zero indicates better performance of the algorithm and close to more than one indicates worst performance of the algorithm. In this work, our objective is to find the global minimum. Hence, lower the '*minimum*', '*mean*' and '*std.*', better is the algorithm. In this work, the tuning parameters of GA are: crossover probability = 70% and mutation probability = 20% and α = 0.01, β_0 = 1 and γ = 1 are tuning parameters for FA and tuning parameters for DE and ADE are control parameter (F) = 1, Crossover Rate (CR) = 10, and strategy = 2. Form Table 1, for Ackely function ADE minimum value is 3.6358e-07 which is 0.0263, 1.2898, 4.6842e-07 less than the GA, FA and DE respectively. With this example we can conclude that ADE is better than other algorithms as its minimum value is near to the theoretical value. The performance of the ADE algorithm is better than the GA, FA and DE, but for some benchmark functions the performance ADE is not so differentiable. For the benchmark functions Bird, Chichinadze, Goldsteinprice, Schweffel, Testtubeholder and Zettl the performance of ADE almost all similar to GA, FA and DE. Table. 2 shows the performance of ADE against GA, FA and DE on five benchmark functions with no change in dimension and population, but the number of iterations is reduced to 100. Even with lesser number of iterations ADE out perform the GA, FA and DE.

Table 3, 4 and 5 shows the performance of ADE against GA, FA and DE with 50 population, 100 iterations and dimensions are 30, 60 and 100 respectively on three benchmark functions. These tables show that even higher dimensions of the problem ADE performance better than the GA, FA and DE. It is concluded that ADE algorithm outperforms the other algorithms in low dimensional and high dimensional search space.

Table 2. *Dimension = 2, population = 50, iterations = 1000.*

Function	Measure	GA	FA	DE	ADE
Akely	minimum	1.0536	0.0022	2.98E-04	2.21E-04
(-4, 5)	mean	1.0536	0.0196	0.0019	6.22E-04
Global min = 0	std.	0	0.0113	0.0013	2.71E-04
Powell	minimum	11.5801	0.0041	0.0467	0.001
(-4, 5)	mean	11.5801	0.0216	0.2286	0.0355
Global min = 0	std.	0	0.0186	0.2714	0.0343
Beale	minimum	0.0077	4.62E-05	3.88E-05	2.24E-05
(-4.5, 4.5)	mean	0.0077	0.0764	0.0019	0.0014
Global min = 0	std.	0	0.2409	0.0016	0.0015
Chichinadze	minimum	-42.9244	-42.9415	-42.9434	-42.9443
(-30, 30)	mean	-42.5205	-42.8421	-42.7111	-42.8863
Global min = -43.31	std.	0.1537	0.1823	0.2137	0.1378
testtubeholder	minimum	-10.8719	-10.8717	-10.8718	-10.8723
(-10, 10)	mean	-10.7198	-10.8657	-10.8635	-10.8676
Global min = -10.87	std.	0.2083	0.0087	0.0092	0.0061

Table 3. *Dimension =30, population =50, iterations = 100.*

Function	Measure	GA	FA	DE	ADE
Akely	minimum	4.4264	1.4553	7.6440	7.4363
(-4, 5)	mean	5.2181	1.7013	8.0370	7.8702
Global min = 0	std.	0.4574	0.1730	0.2483	0.2587
Exponential	minimum	1.7131	1.4553	0.6440	0.4363
(-4, 5)	mean	0.5308	1.7013	0.0370	0.8702
Global min = 0	std.	0.1182	0.1730	0.2483	0.2587
Brown	minimum	64.2397	21.0185	7.2463	6.4224
(-4, 5)	mean	428.1775	21.5115	7.4442	5.5335
Global min = 0	std.	847.7448	20.2839	2.0540	4.0634

Table 4. *Dimension =60, population =50, iterations = 100.*

Function	Measure	GA	FA	DE	ADE
Akely	minimum	4.4264	1.4553	7.644	3.4363
(-4, 5)	mean	5.2181	1.7013	8.037	3.8702
Global min = 0	std.	0.4574	0.173	0.2483	0.2587
Exponential	minimum	-0.4046	-0.8453	3.3118	3.1224
(-4, 5)	mean	-0.2614	-0.7983	3.5443	3.5335
Global min = 0	std.	0.0825	0.0329	0.1051	0.0634
Brown	minimum	186.1389	59.5164	36.601	34.232
(-4, 5)	mean	985.7908	60.5707	47.61	35.344
Global min = 0	std.	1.49E+03	70.7381	14.919	13.121

Table 5. *Dimension =100, population =50, iterations = 100.*

Function	Measure	GA	FA	DE	ADE
Akely	minimum	15.2839	12.6322	8.9616	7.0844
(-4, 5)	mean	15.7639	12.7814	9.2831	7.1948
Global min = 0	std.	0.4142	0.0741	0.1338	0.0876
Exponential	minimum	-0.1214	-0.6046	3.5954	3.4771
(-4, 5)	mean	-0.0698	-0.5453	3.6534	3.6021
Global min = 0	std.	0.0342	0.0482	0.0355	0.0711
Brown	minimum	1.36E+03	144.5975	39.05	11.893
(-4, 5)	mean	5.48E+07	161.4384	40.089	13.021
Global min= 0	std.	1.73E+08	123.4041	52.115	10.711

The simulation results of six benchmark functions, *i.e*, Akely, Bukin4, Schweffel, Himmelbalu, Crosslegtable, and Zettle are shown in Fig. 7. These graphs are drawn between the number of iterations and the corresponding objective function value with 100 function evaluations. From these Figs., it is observed that ADE time of convergence is better to GA, FA and DE. Form Fig. 7a, one can compare performances of all four algorithms. Note that six figures (Figs. 7a-7e) are displayed for six benchmark functions separately. From Fig. 7, it is seen that our proposed algorithm ADE outperforms all other algorithms. So the proposed ADE is further used for codebook design in cascaded pattern search. The flow chart of proposed hybrid pattern search and ADE approach is shown in Fig. 8.

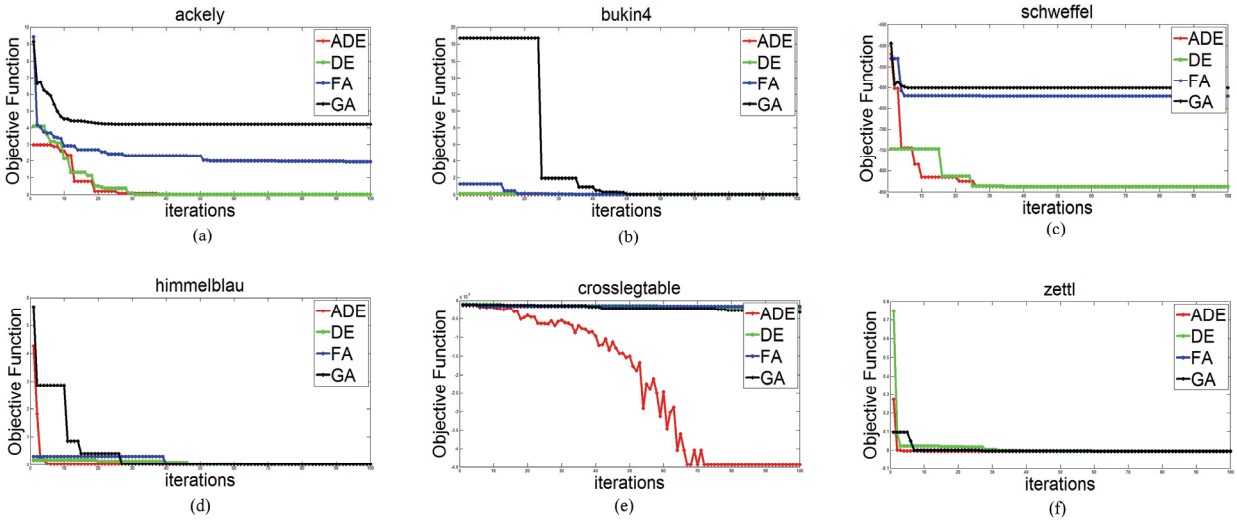

Fig. 7. *Performance of proposed ADE, DE, GA and FA for P = 50 and D = 30.*

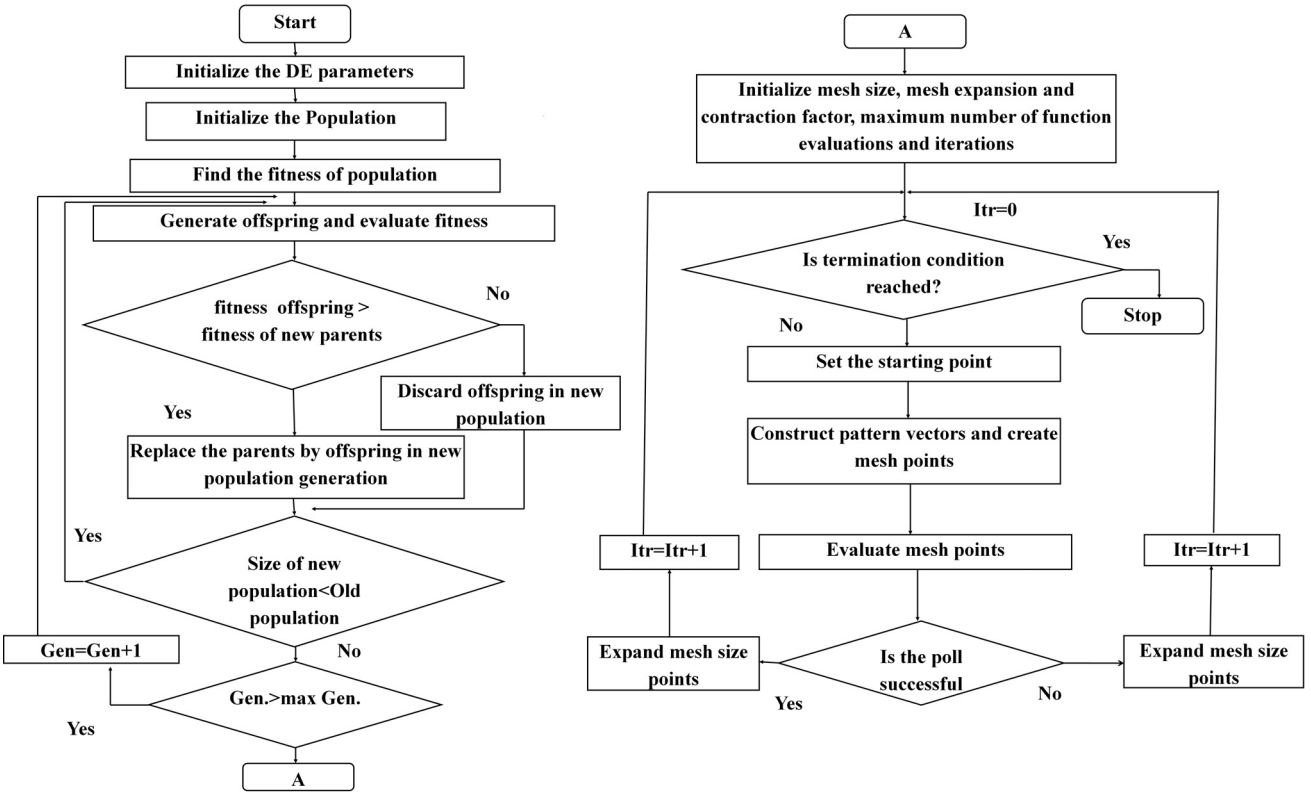

Fig. 8. *Flowchart of proposed hADE-PS.*

RESULTS

For the validation of the proposed hybrid ADE-PS method, four standard SAR images from the image database of compression Sandia National Laboratories 'flight line' and 'baseball diamond' and all the images are grayscale of size 256 by 256 stored in .jpg format are considered. The hybrid ADE-PS and other algorithms, *i.e.*, K-Means, DE and ADE have been implemented in Matlab 7.9.0 (R2009b) on HP Laptop, Intel Core i5 processor 4GB RAM and 32 bit operating system. All the optimization algorithms acquired the K-Means codebook as the preliminary codebook or one of the population/solution. The parameters used for comparison of proposed hADE-PS algorithm with others are bitrate/bits per pixel, PSNR, Mean Square Error (MSE), SSIM, fitness function, compression ratio and memory size of output image.

Bitrate/Bits Per Pixel (BPP) & Compression Ratio: Bitrate is the indication of number bits used to represent the vector quantized image, In this work two kinds of bitrates and compression ratio are defined; one before the run-length followed huffman coding and another after run-length followed Huffman coding. The formula for the bitrate and compression ratio of the first one (*BPP & CR*) and second one (BPP_{code} & CR_{code}) is given in Eq. 9 and Eq. 10 respectively.

$$\mathrm{BPP} = \frac{\log_2 N_c}{k} \qquad \mathrm{CR} = \frac{8 \times k}{N_c}, \tag{9}$$

where N_c is codebook size and k is non-overlapping image block size (4*4)

$$bpp = \frac{total\ number\ of\ bits\ transmitted}{size\ of\ image}$$
$$CR_{code} = \frac{size\ of\ image * 8}{total\ number\ of\ bits\ transmitted} \tag{10}$$

Peak Signal to Noise Ratio *(PSNR)*: It's a measure of reconstructed/decompressed image quality which is given in Eq. 11

$$PSNR = 10 \times \log_{10}\left(\frac{255^2}{MSE}\right)(\mathrm{dB}), \tag{11}$$

where Mean Square Error (*MSE*) which is given in Eq. 12

$$MSE = \frac{1}{M \times N}\sum_{I}^{M}\sum_{J}^{N}\{f(\mathrm{I},\mathrm{J}) - \overline{f}(\mathrm{I},\mathrm{J})\}^2, \tag{12}$$

where $M \times N$ is the size of the image, I and J represents the pixel value of original and decompressed images. In our experiment we have taken $N=M$ a square image. $f(I,J)$ is an original image and $\overline{f}(I,J)$ reconstructed image of size 256 by 256.

Structural Similarity Index Measure (*SSIM*): It evaluates the visual similarity between the original image and the reconstructed image. High-quality image is one whose structure closely matches that of the original input image. The structural similarity index is calculated between original input image and reconstructed image given in Eq. 13.

$$SSIM = \frac{(2\mu_I\mu_{\tilde{I}} + C_1)(2\sigma_{I\tilde{I}} + C_2)}{(\mu_I^2 + \mu_{\tilde{I}}^2 - C_1)(\sigma_I^2 + \sigma_{\tilde{I}}^2 - C_2)}, \quad (13)$$

where μ_I and $\mu_{\tilde{I}}$ are the mean value of the original image I and reconstructed image $\tilde{I}$, σ_I and $\sigma_{\tilde{I}}$ are the standard deviation of original image I and reconstructed image $\tilde{I}$, $\sigma_{I\tilde{I}}$ is the cross-correlation and C_1 & C_2 are constants are equal to 0.065.

$$\text{Mean value} = \mu_I = \frac{1}{N}\sum_{i=1}^{N} I_i \,. \quad (14)$$

Standard deviation =

$$\sigma_{I\tilde{I}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{I}_i - \mu_I)(\tilde{\mathrm{I}}_i - \mu_{\tilde{I}}) \,. \quad (15)$$

DISCUSSIONS

The proposed hADE-PS based vector quantization is compared with the standard K-Means, Differential Evolution and Adaptive Differential evolution. The proposed method is evaluated with a codebook size of 8, 16, 32, 64, 128, 256 and 512 and each algorithm runs five times. It is observed that PSNR value is increased with increment in codebook size. The same parameters are used for all experiments (for consistency).

Fig. 9 to Fig. 12 shows the graph between BPP to PSNR and bitrate to the fitness function. In the proposed work fitness function is the ratio of sum of the Euclidean distance between the input image and codewords (distortion) to size of the input image. So motto is to design or optimize codebook which minimizes distortion. From the figures, it can be observed that the peak signal to noise ratio of hADE-PS is superior to the K-Means, DE and ADE. The PSNR value of hADE-PS is around 0.1 to 0.3 higher than the K-Means, DE and ADE because of its exploration and exploitation behavior. This 0.1 to 0.3 is not a huge difference, but its reflection on MSE between the input image and the reconstructed image is very huge (around 10 times lesser). This is specified in Fig. 13 to Fig. 16, where a difference of 0.1 in PSNR between ADE and hADE-PS is equal to the difference in MSE of around 283099 for flight line image. Fig. 13 to Fig. 16 shows the bar chart of MSE of four methods for four images and observed MSE is decreasing with the increment in codebook size and MSE of hADE-PS is considerable smaller than the K-Means, DE and ADE.

From Table 6 and 7, it is observed that the proposed hADE-PS method achieved lesser bits per pixel at an average of 0.77% and 0.56% as compared to DE and ADE and higher at an average of 2.77% as compared to K-Means. hADE-PS achieved higher compression ratio at an average of 5.6% and 4.4% compared to DE and ADE and lesser at an average of 1.24% compared to K-Means for Baseball SAR image. Whereas hADE-PS method achieved higher SSIM for Baseball SAR image at an average of 0.52% compared to K-Means and lesser at an average of 1.50% and 1.36% compared to DE and ADE respectively.

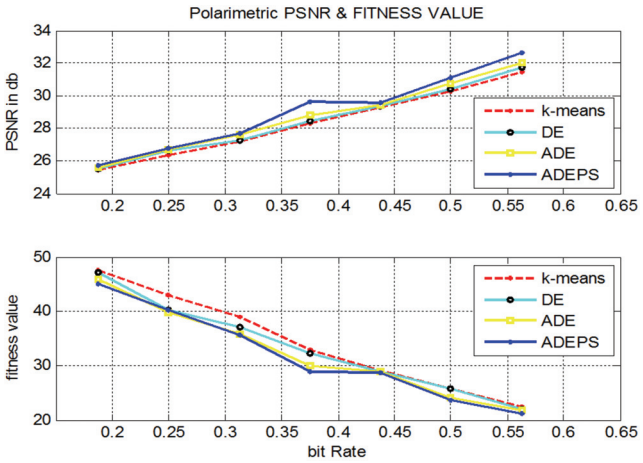

Fig. 9. *PSNR & fitness function of Polarimetric SAR image.*

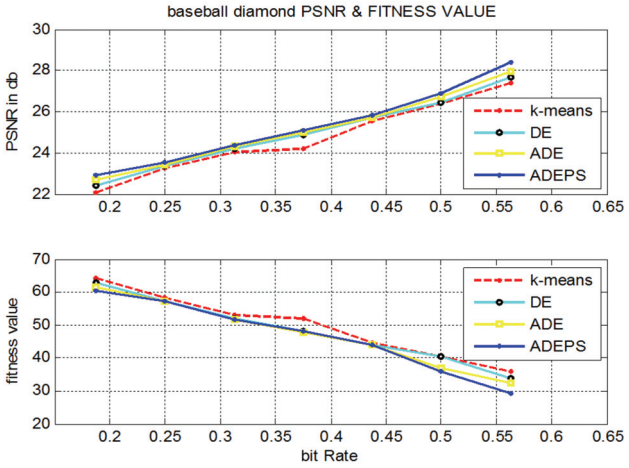

Fig. 10. *PSNR & fitness function of BaseBall Diamond SAR image.*

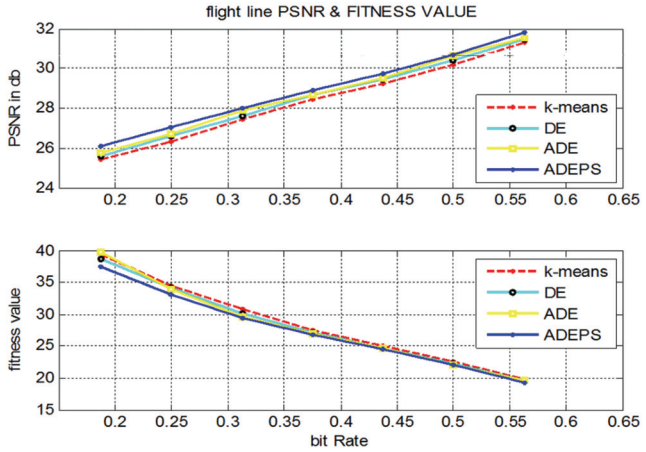

Fig. 11. *PSNR & fitness function of Flight Line SAR image.*

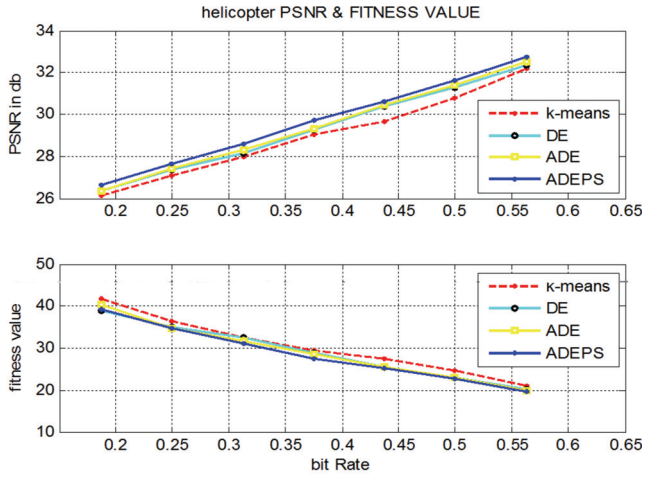

Fig. 12. *PSNR & fitness function of Flight Line SAR image.*

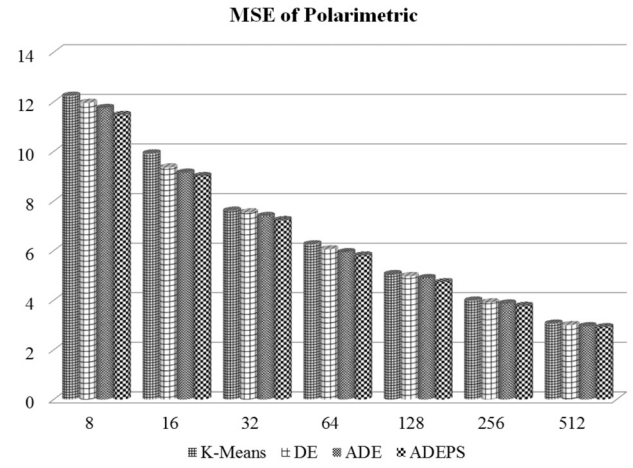

Fig. 13. *Polarimetric image MSE with a codebook size of 8, 16, 32, 64, 128, 256 and 512.*

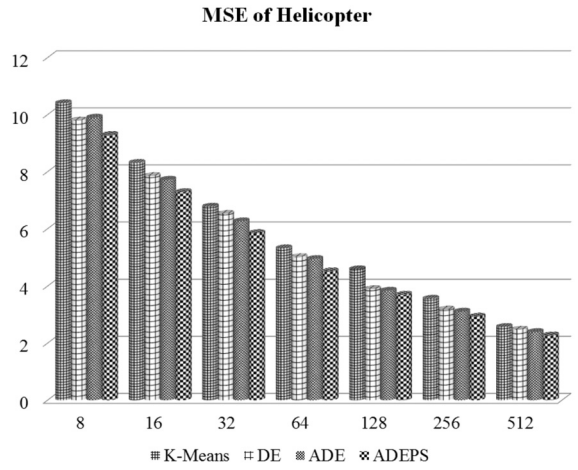

Fig. 14. *Helicopter image MSE with a codebook size of 8, 16, 32, 64, 128, 256 and 512.*

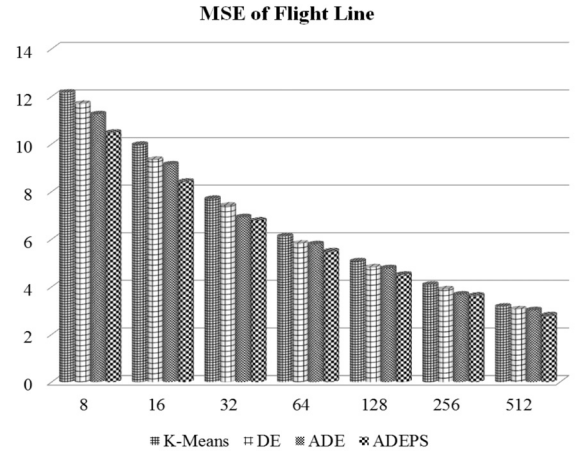

Fig. 15. *Flight Line image MSE with a codebook size of 8, 16, 32, 64, 128, 256 and 512.*

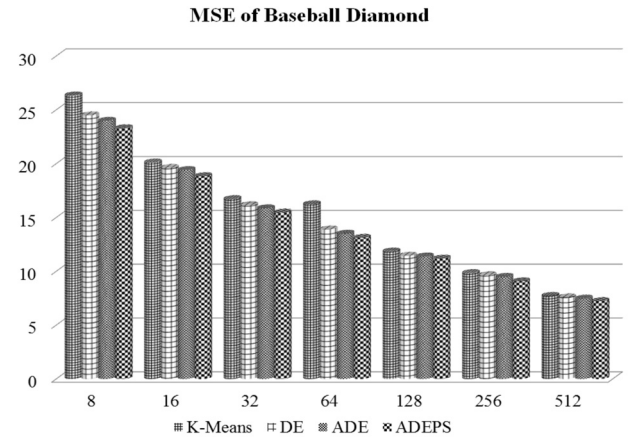

Fig. 16. *Baseball Diamond image MSE with a code book size of 8, 16, 32, 64, 128, 256 and 512.*

From Table 8 and 9, it is observed that the proposed hADE-PS method achieved lesser bits per pixel at an average of 0.27% and 1.15% compared to K-Means and DE and higher at an average of 0.41% compared to ADE and achieved higher compression ratio at an average of 0.8% and 2.6% compared to K-Means and DE and lesser at an average of 7.09% compared to ADE for Flight line SAR image. Whereas hADE-PS method achieved higher SSIM at an average of 1.02%, 0.03% and 1.28% compared to K-Means, DE and ADE respectively for Flight line SAR image.

From Table. 10 and 11, it is observed that the proposed hADE-PS method achieved lesser bits per Pixelat an average of 1.65%, 0.80% and 0.90% with compare to K-Means, DE and ADE respectively, and hADE-PS method achieved higher compression ratio at an average of 7.57%, 5.29% and 6.06% with compare to K-Means, DE and ADE respectively for Helicopter SAR image. Whereas hADE-PS method achieved higher SSIM at an average of 0.13%, and 0.29% compared to K-Means and DE respectively, and lesser at an average of 0.64% compared to ADE.

From Table. 12 and 13, it is observed that the proposed hADE-PS method achieved higher bits per Pixelat an average of 1.17% and 0.3% compared to K-Means and DE and no difference compared to ADE and hADE-PS method achieved higher compression ratio at an average of 0.33% compared to ADE and lesser at an average of 4.09% and 1.88% compared to K-Means and DE Polarimetric SAR image. Whereas hADE-PS method achieved higher SSIM at an average of 1.06%, 0.56% and 0.41% compared to K-Means, DE and ADE respectively. From Table 6-13 it is observed that a lower and lower value of minimum value of output images and a higher and higher value of maximum value of output image shows the better PSNR and SSIM.

Table 6. *The bits per pixel (BPP), Compression ratio (CR), Structural similarity index (SSIM), Number of bits for codebook (bcodebook) and index (bindex), min & max of output image and the number of bytes (bout) of Baseball images by using the four different algorithms with different codebook size (NC).*

N_C	Technique	BPP_{code}	CR_{code}	SSIM	$b_{codebook}$	b_{index}	min & max	b_{out}
8	K-Means	0.145142	55.11859	0.315150	680	8832	25,157	1189
	DE	0.159668	50.10398	0.342557	728	9736	31175	1308
	ADE	0.153564	52.09539	0.324821	688	9376	34,168	1258
	ADEPS	0.143555	55.72789	0.308223	728	8680	37,179	1176
16	K-Means	0.222778	35.91014	0.431154	1600	13000	26,179	1825
	DE	0.215332	37.15193	0.416324	1616	12496	25,179	1764
	ADE	0.217651	36.75603	0.411986	1624	12640	25,178	1783
	ADEPS	0.201172	39.76699	0.405823	1640	11544	27,180	1648
32	K-Means	0.299316	26.72757	0.493829	3320	16296	25,181	2452
	DE	0.307373	26.02701	0.496363	3408	16736	19,186	2518
	ADE	0.304443	26.27747	0.499020	3384	16568	23,184	2494
	ADEPS	0.297607	26.88105	0.486835	3352	16152	21,191	2438
64	K-Means	0.311768	25.66014	0.499794	3376	17056	24,187	2554
	DE	0.430664	18.57596	0.566458	6880	21344	19,196	3528
	ADE	0.429565	18.62347	0.574566	6896	21256	18,198	3519
	ADEPS	0.420776	19.01247	0.565834	7040	20536	9,250	3447
128	K-Means	0.613525	13.03940	0.637223	13800	26408	16,205	5026
	DE	0.611938	13.07321	0.637296	13872	26232	16,218	5013
	ADE	0.609863	13.11769	0.639557	13856	26112	16,212	4996
	ADEPS	0.607056	13.17836	0.635659	14048	25736	8,222	4973
256	K-Means	0.884277	9.046935	0.704460	27336	30616	16,222	7244
	DE	0.890137	8.987383	0.702164	27712	30624	12,231	7292
	ADE	0.890747	8.981225	0.703864	27680	30696	16,223	7297
	ADEPS	0.898804	8.900720	0.704363	28208	30696	14,243	7363
512	K-Means	1.379639	5.798620	0.767407	55264	35152	7,242	11302
	DE	1.382202	5.787865	0.766288	55464	35120	4,250	11323
	ADE	1.383301	5.783269	0.768358	55552	35104	10,237	11332
	ADEPS	1.397705	5.723668	0.762571	56576	35024	9,243	11450

Table 7. *Average BPP_{code}, CR_{code}, SSIM and % difference against ADEPS of Baseball SAR image.*

	BPP_{code}	% difference against ADEPS	CR_{code}	% difference against ADEPS	SSIM	% difference against ADEPS
K-Means	0.550921	2.778876	24.47163	-1.24728	0.549860	0.52428
DE	0.571045	-0.77241	22.81533	5.605383	0.561064	-1.50265
ADE	0.569876	-0.56612	23.09065	4.466292	0.560310	-1.36624
ADEPS	0.566668		24.17016		0.552758	

Table 8. *The bits per pixel (BPP), Compression ratio (CR), Structural similarity index (SSIM), Number of bits for codebook (bcodebook) and index (bindex), min & max of output image and the number of bytes (bout) of Flight line images by using the four different algorithms with different codebook size (NC).*

N_C	Technique	BPP_{code}	CR_{code}	SSIM	$b_{codebook}$	b_{index}	min & max	b_{out}
8	K-Means	0.114014	70.16702	0.220490	600	6872	10,102	934
	DE	0.116699	68.55230	0.225755	560	7088	15,113	956
	ADE	0.097290	82.22836	0.198176	560	5816	18,116	797
	ADEPS	0.112427	71.15744	0.226754	600	6768	16,155	921
16	K-Means	0.184326	43.40132	0.304027	1400	10680	7,116	1510
	DE	0.188843	42.36328	0.312903	1352	11024	8,131	1547
	ADE	0.168945	47.35260	0.290382	1376	9696	6,160	1384
	ADEPS	0.176636	45.29095	0.294902	1408	10168	9,180	1447
32	K-Means	0.260498	30.71040	0.366904	2944	14128	7,147	2134
	DE	0.264404	30.25669	0.366673	2984	14344	6,160	2166
	ADE	0.258911	30.89863	0.371584	3016	13952	6,161	2121
	ADEPS	0.267822	29.87056	0.366819	3088	14464	6,184	2194
64	K-Means	0.386108	20.71957	0.444107	6200	19104	4,174	3163
	DE	0.383301	20.87134	0.439294	6360	18760	3,185	3140
	ADE	0.389404	20.54420	0.445250	6288	19232	5,180	3190
	ADEPS	0.381470	20.97152	0.434910	6456	18544	6,189	3125
128	K-Means	0.559692	14.29357	0.515133	12544	24136	3,187	4585
	DE	0.571899	13.98847	0.527041	12808	24672	3,197	4685
	ADE	0.571167	14.00641	0.522511	13064	24368	2,187	4679
	ADEPS	0.574829	13.91718	0.521636	13400	24272	4,196	4709
256	K-Means	0.855713	9.348930	0.605669	26096	29984	2,202	7010
	DE	0.866699	9.230423	0.614450	26656	30144	2,215	7100
	ADE	0.859497	9.307769	0.607158	26720	29608	3,227	7041
	ADEPS	0.861938	9.281405	0.604545	26800	29688	2,235	7061
512	K-Means	1.338623	5.976290	0.694850	52808	34920	0,202	10966
	DE	1.344360	5.9507850	0.692779	53296	34808	0,211	11013
	ADE	1.344238	5.9513260	0.694665	53248	34848	1,211	11012
	ADEPS	1.362305	5.8724010	0.693082	54416	34864	0,234	11160

Table 9. *Average BPP_{code}, CR_{code}, SSIM and % difference against ADEPS of Flight line SAR image.*

	BPP_{code}	% difference against ADEPS	CR_{code}	% difference against ADEPS	SSIM	% difference against ADEPS
K-Means	0.450169	-0.27152	27.80244	0.888362	0.528425	1.028810
DE	0.454128	-1.15336	27.31618	2.621807	0.533744	0.032589
ADE	0.447104	0.411182	30.04133	-7.09295	0.527065	1.283530
ADEPS	0.448950		28.05164		0.533918	

Table 10. *The bits per pixel (BPP), Compression ratio (CR), Structural similarity index (SSIM), Number of bits for codebook ($b_{codebook}$) and index (b_{index}), min & max of output image and the number of bytes (b_{out}) of Helicopter images by using the four different algorithms with different codebook size (N_C).*

N_C	Technique	BPP_{code}	CR_{code}	SSIM	$b_{codebook}$	b_{index}	min & max	b_{out}
8	K-Means	0.168213	47.55878	0.331316	688	10336	11,78	1,378
	DE	0.166015	48.18800	0.320182	688	10192	11,77	1360
	ADE	0.173462	46.11963	0.342659	672	10696	6,85	1,421
	ADEPS	0.138306	57.84289	0.317535	640	8424	11,78	1,133
16	K-Means	0.235474	33.97408	0.407295	1416	14016	10,94	1,929
	DE	0.220703	36.24779	0.399017	1456	13008	5,83	1808
	ADE	0.216919	36.88014	0.405095	1424	12792	3,78	1,777
	ADEPS	0.227783	35.12111	0.409797	1544	13384	5,191	1,866
32	K-Means	0.318970	25.08075	0.476180	2984	17920	6,88	2613
	DE	0.312500	25.60000	0.472684	3048	17432	6,11	2560
	ADE	0.305298	26.20392	0.467926	3088	16920	7,80	2501
	ADEPS	0.315918	25.32303	0.479451	3096	17608	3,203	2588
64	K-Means	0.428101	18.68720	0.541240	6096	21960	5,177	3507
	DE	0.423218	18.90280	0.532136	6240	21496	3,173	3467
	ADE	0.433105	18.47125	0.547188	6272	22112	3,197	3548
	ADEPS	0.409546	19.53383	0.529461	6344	20496	3,203	3355
128	K-Means	0.601807	13.29331	0.606344	12680	26760	2,188	4930
	DE	0.592773	13.49588	0.612615	12568	26280	2,203	4856
	ADE	0.595093	13.44328	0.609848	12792	26208	1,203	4875
	ADEPS	0.597290	13.39383	0.611974	26136	26136	1,253	4893
256	K-Means	0.858643	9.317032	0.673415	25072	31200	1,197	7034
	DE	0.862427	9.276150	0.684333	25528	30992	2,203	7065
	ADE	0.860962	9.291933	0.683863	25416	31008	1,203	7053
	ADEPS	0.860107	9.301164	0.683592	25656	30712	1,253	7046
512	K-Means	1.318237	6.068710	0.744965	51048	35344	0,255	10799
	DE	1.318970	6.065340	0.753700	51152	35288	0,203	10805
	ADE	1.315918	6.079406	0.753683	50984	35256	0,253	10780
	ADEPS	1.316650	6.076024	0.754213	51144	35144	0,255	10786

Table 11. *Average BPP_{code}, CR_{code}, SSIM and % difference against ADEPS of Helicopter SAR image.*

	BPP_{code}	% difference against ADEPS	CR_{code}	% difference against ADEPS	SSIM	% difference against ADEPS
K-Means	0.561349	-1.65162	21.99712	7.570621	0.540108	0.139038
DE	0.556658	-0.8021	22.53942	5.291939	0.539238	0.299893
ADE	0.557251	-0.90948	22.35565	6.064119	0.544323	-0.64028
ADEPS	0.552228		23.79884		0.540860	

Fig. 17 to Fig. 20 shows the reconstructed images of 'Baseball diamond' 'Flight line', 'Helicopter', and 'Polarimetric', with a codebook size of 64, 16, 8 and 8 respectively for K-Means, DE, ADE and hADE-PS. These codebook size is selected only for the better explanation and visual proof of the proposed method. For the best explanation of stated method, the regions marked with the red pencil shows reconstructed image visible clearly with the proposed method. For flight line image the regions marked with red are smoother with hADE-PS, whereas with other methods these regions are blurred as shown in Fig. 18. Similarly for Baseball diamond, Helicopter and Polarimetric the marked red regions are smooth with hADE-PS compared to other K-Means, DE, ADE optimization techniques.

Table 12. *The bits per pixel (BPP), Compression ratio (CR), Structural similarity index (SSIM), Number of bits for codebook ($b_{codebook}$) and index (b_{index}), min & max of output image and the number of bytes (b_{out}) of Polarimetric images by using the four different algorithms with different codebook size (N_C).*

N_C	Technique	BPP_{code}	CR_{code}	SSIM	$b_{codebook}$	b_{index}	min & max	b_{out}
8	K-Means	0.153931	51.97145	0.383693	712	9376	27,122	1261
	DE	0.154907	51.64381	0.384900	704	9448	26,121	1269
	ADE	0.166870	47.94148	0.393032	720	10216	24,120	1367
	ADEPS	0.167725	47.69723	0.399023	704	10288	18,120	1374
16	K-Means	0.220825	36.22775	0.473461	1560	12912	24,130	1809
	DE	0.236206	33.86873	0.488854	1520	13960	3,120	1935
	ADE	0.234009	34.18675	0.483875	1480	13856	5,124	1917
	ADEPS	0.231567	34.54718	0.492617	1536	13640	17,124	1897
32	K-Means	0.318481	25.11920	0.564017	3184	17688	3,141	2609
	DE	0.321899	24.85248	0.568047	3104	17992	14,133	2637
	ADE	0.321655	24.87135	0.569604	3240	17840	9,127	2635
	ADEPS	0.315918	25.32303	0.564702	3216	17488	7,144	2588
64	K-Means	0.424194	18.85928	0.631893	6320	21480	0,167	3475
	DE	0.432495	18.49732	0.635423	6544	21800	2,184	3543
	ADE	0.431763	18.52870	0.630889	6528	21768	2,147	3537
	ADEPS	0.433838	18.44007	0.639282	6688	21744	2,191	3554
128	K-Means	0.598145	13.37469	0.699715	13104	26096	1,155	4900
	DE	0.601929	13.29061	0.699861	13376	26072	1,175	4931
	ADE	0.606079	13.19960	0.708086	13408	26312	1,162	4965
	ADEPS	0.601929	13.29061	0.701873	13488	25960	1,175	4931
256	K-Means	0.872681	9.167156	0.759742	26576	30616	0,176	7149
	DE	0.870972	9.185144	0.757715	26680	30400	1,179	7135
	ADE	0.871582	9.178711	0.756357	26744	30376	1,191	7140
	ADEPS	0.874756	9.145409	0.762382	27016	30312	1,187	7166
512	K-Means	1.349731	5.927105	0.815328	53480	34976	1,182	11057
	DE	1.354370	5.906805	0.814673	53752	35008	0,179	11095
	ADE	1.352783	5.913734	0.814175	53664	34992	0,190	11082
	ADEPS	1.359009	5.886643	0.814358	54096	34968	0,179	11133

Table 13. *Average BPP_{code}, CR_{code}, SSIM and % difference against ADEPS of Polarimetric SAR image.*

	BPP_{code}	% difference against ADEPS	CR_{code}	% difference against ADEPS	SSIM	% difference against ADEPS
K-Means	0.562570	1.17330	22.94952	-4.09282	0.618264	1.060505
DE	0.567540	0.30022	22.46356	-1.88863	0.621353	0.566179
ADE	0.569249	0	21.97433	0.330383	0.622288	0.416553
ADEPS	0.569249		22.04717		0.624891	

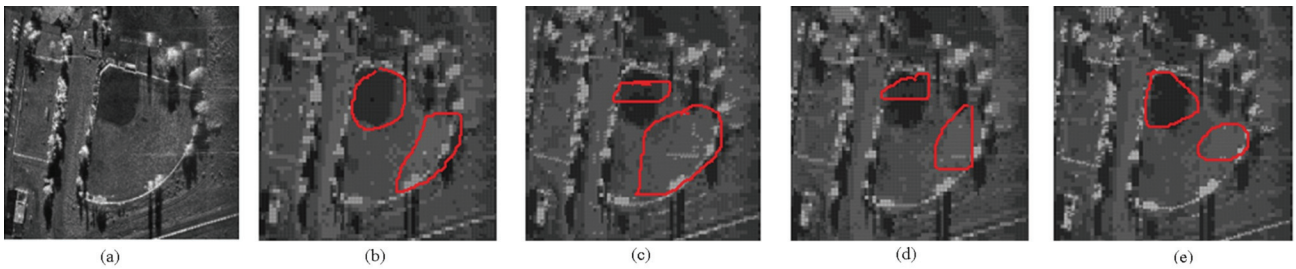

Fig. 17. *Reconstructed Baseball image with a codebook size of 64 (a) Input Image (b) K-Means (c) DE (d) ADE (e) hADE-PS.*

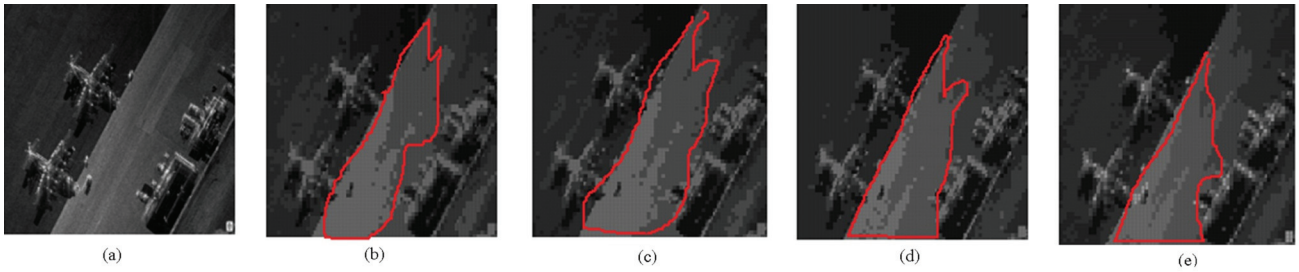

Fig. 18. *Reconstructed Flight line SAR image with a codebook size of 16 (a) Input Image (b) K-Means (c) DE (d) ADE (e) hADE-PS.*

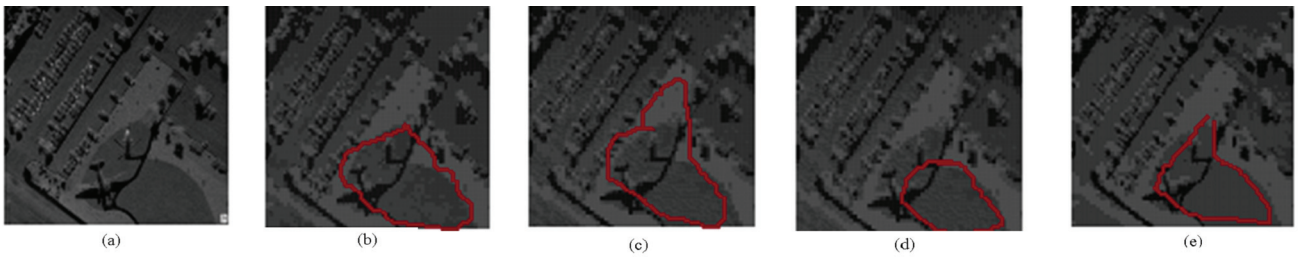

Fig. 19. *Reconstructed Helicopter SAR image with a codebook size of 8 (a) Input Image (b) K-Means (c) DE (d) ADE (e) hADE-PS.*

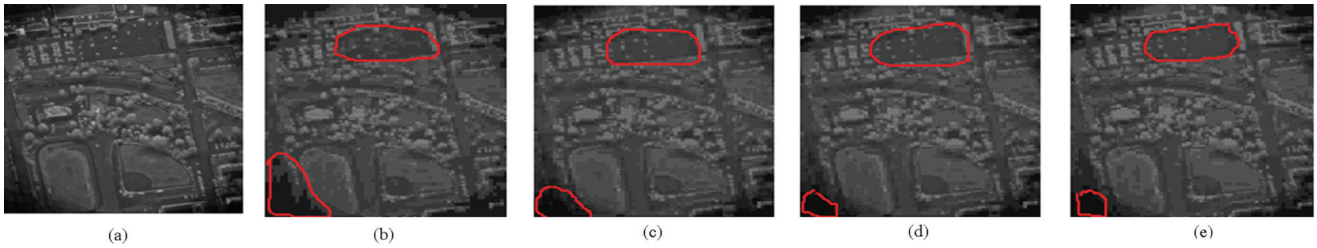

Fig. 20. *Reconstructed Polarimetric SAR image with a codebook size of 8 (a) Input Image (b) K-Means (c) DE (d) ADE (e) hADE-PS.*

CONCLUSIONS

In this paper, the performance analysis of different optimization techniques is compared which are used for optimizing the LBG vector quantization for efficient codebook design, that results in high image compression and good reconstructed image quality. Experimental results shows that hADE-PS has better performance in compression ratio and reconstructed image quality than DE, ADE and K-Means. Adaptive Differential Evolution algorithm gives true global minimum regardless of the initial parameter values with fast convergence, and with few control parameters.

REFERENCES

Abouali AH (2015). Object-based VQ for image compression. Ain Shams Engineering Journal, 6:211-16.

Chen Q, Yang JG, Gou J (2005). Image Compression Method by using Improved PSO Vector Quantization. Advances in Natural Computation, intr conf on neural computation (ICNC 2005), 490-95.

Chen Q (2012). Vector quantization method for image compression based on GA and LBG clustering algorithm. Computer Science 18:213–21.

Chiranjeevi K, Umaranjan J (2015). Fast vector quantization using a Bat algorithm for image compression. Engg Science and Tech, an Inter J 39:769–81.

Daubechies I (1988). Orthonormal basis of compactly supported wavelets. Comm. Pure Appl. Math., 12:909–96.

George ET, Dimitrios MT (2012). Fuzzy Clustering-Based Vector Quantization for Image Compression. Comp Intel in Image Proc, August, 93–105.

Hooke R, Jeeves TA. Direct search solution of numerical and statistical problems. J Assoc for Compu Machinery 8:212–29.

Horng MH, Jiang TW (2011). Image Vector Quantization Algorithm via Honey Bee Mating Optimization. Expert Sys with App, 38:1382–92.

Horng MH (2012). Vector Quantization using the firefly algorithm for Image Compression. Expert Sys with App 39:1078–91.

Hu YC, Su. BH, Tsou CC (2008). Fast VQ Codebook Search for Gray Scale Image Coding. Image and Vision Comp 26:657–66.

Krishna K, Ramakrishnan, Thathachar M (1997). Vector quantization using genetic k-means algorithm for image compression. Intr Conf on Infor, Comm and Signal Pro 3:1585–87.

Linde Y, Buzo A, Gray RM (1980). An algorithm for vector quantize design. IEEE Tran on Comm 28:84–95.

Liu L, Ling C (2015). Polar Lattices are Good for Lossy Compression. IEEE Info Theory Workshop 342–46.

Patane G, Russo M. (2002). The enhanced LBG algorithm. Neural Networks 14:1219–37.

Poggi G, Ragozini ARP (2001). Tree-structured product-codebook vector quantization. Signal Proc: Img Comm 16:421–30.

Price K, Storn RM, Lampinen J (2005). Differential evolution: A practical approach to global optimization. Natural Comp Ser Berlin: Springer, 732–41.

Rajpoot A, Hussain A, Saleem K, Qureshi Q (2004). A Novel Image Coding Algorithm Using Ant Colony System Vector Quantization. Intr workshop on sys, sig and img proc, Poznan, Poland 2:812-23.

Sanyal N, Chatterjee A, Munshi (2013). Modified bacterial foraging optimization technique for vector quantization-based image compression. Comp Int in Img Proc, Springer, Berlin, Heidelberg, 2:131–52.

Stron R, Price K (1995). Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. J of Global Optimize 11:341–59.

Tsai CW, Tseng SP, Yang CS, Chiang MC (2013). PREACO: A Fast Ant Colony Optimization for Code-book Generation. Applied Soft Computing 13:3008–20.

Wang Y, Feng XY, Huang YX, Pu DB, Zhou WG, Liang YC (2007). A Novel Quantum Swarm Evolutionary Algorithm and its Applications. Neurocomputing 70: 633–40.

Yao X, Liu Y, Lin G (1999). Evolutionary programming made faster. IEEE Tran on Evol Compu 3:82–102.

Zeng Z, Cumming IG (2001). SAR Image Data Compression Using a Tree-Structured Wavelet Transform. IEEE Tran on Geosci and Remote Sen 39:546–52.

Zhao M, Liu H (2013). Vector quantization codebook design and application based on the clonal selection algorithm. Sensors and Transducers 159:415–21.

Zheng X, Julstrom BA, Cheng W (1997). Design of Vector Quantization Codebooks Using a Genetic Algorithm. IEEE Intr Conf on Evolutionary Comp, 13-16 April, University Place Hotel Indianapolis, USA, 525–29.