Journal of Information Hiding and Multimedia Signal Processing ©2016 ISSN 2073-4212
Ubiquitous International Volume **7**, Number **5**, September **2016**

A Novel Pixel Grouping Scheme for AMBTC Based on Particle Swarm Optimization

Zhihong Li[1], Qiang Jin[1], Chin-Chen Chang[2,*], Anhong Wang[1], Li Liu[1]

[1]Institute of Digital Multimedia and Communication,
Taiyuan University of Science and Technology, Taiyuan 030024, China.

[2,*]Department of Information Engineering and Computer Science,
Feng Chia University, Taichung 40724, Taiwan.
∗Correspondence Autor
alan3c@gmail.com

Received August, 2015; revised June, 2016

Abstract. *Block truncation coding (BTC) is an efficient compression method for grayscale images. Based on the basic concept of traditional BTC, Absolute moment block truncation coding (AMBTC), which utilizes the average value of each block as the threshold to divide each block into two pixel groups and a corresponding bitmap, is widely used because it is simple and efficient. However, the average value of each block value is not always the best threshold to recover the original image. In this paper, we use particle swarm optimization (PSO) to search the optimal threshold in order to minimize the distortion of the recovered image. In the meantime, by using PSO algorithm, the speed of searching the optimal threshold can be flexibly changed according to two coefficients, the population of initialized particles and the maximum iteration time of PSO. The experimental results show that the restored image of our method has better visual quality than AMBTC and is especially efficient to find the near optimal threshold when the block size is large.*

Keywords: Image compression; AMBTC; Optimal threshold; PSO

1. **Introduction.** With the rapid development of the Internet, digital images are widely used in many fields. However, the large amount of data needed to represent digital images makes its transmission slow and storage expensive. Therefore, it is necessary to compress these images before they are stored or transmitted. There are two branches of image compression, lossless compression [1, 2] and lossy compression [3, 4, 5, 6]. As one of lossy compression techniques, BTC proposed by Delp and Mitchell in 1979 [7], is famous for its efficiency because it has low computation complexity. Moreover, it can maintain a good visual quality of the restored image. Owning to these advantages, BTC has been employed in many fields, such as data hiding [8], edge detection [9], image retrieval [10, 11], and secret sharing [12, 13].

In the process of BTC, the image is first divided into many non-overlapping blocks. Then the first two sample moments of each block are preserved in order to divide this block into two groups. Considering the average value of each block as a threshold, a corresponding bitmap is generated. Finally, the average value of this block, the standard deviation of the block and a corresponding bitmap can be used to reconstruct the original image block.

Many approaches have endeavored to improve the performance of BTC. The first category of these schemes is focusing on preserving the moment characteristic of the original image. In 1984, Halverson et al. [14] generalized a family of moment-preserving quantizers with the potential for improved performance. Later, a modified BTC method [15] was proposed which preserves only the first-order moment. This algorithm is optimal in the mean-square sense for a particular class of BTC algorithms. The second category of these approaches is focused on improving image quality. In 1984, Lema and Mitchell proposed a simpler method named absolute moment block truncation coding (AMBTC) [16]. By preserving two quantization values (i.e., two mean values of the pixel groups) instead of the mean value and the standard deviation of the block, both computing speed and restored image quality are improved. In 1990, Hui [17] proposed an adaptive BTC (ABTC) method by designing a minimum mean square error quantizer for BTC algorithm. Compared with the standard BTC and AMBTC, this method is very efficient and the visual quality of restored image is much better. Also, three-level BTC algorithms [18, 19] have been proposed to enhance the visual quality of restored image.

Both the standard BTC and AMBTC take the average value of each image block as threshold to divide each block into two groups. The use of average value of each block for pixel grouping is very efficiency, but the mean value is not always the best threshold. In other words, the image distortion will be reduced further if we seek out the best threshold for pixel grouping. In 1994, Chen and Liu [20] proposed an optimal BTC (OBTC) method which exhaustively explores all possible values among the block to find the optimal threshold. Although this method can find out the optimal threshold, the computational cost is very expensive. Later, economical BTC (EBTC) [21] was proposed by Yang and Lin in 1996. The goal of EBTC is to reduce the computation complexity of OBTC. However, the optimal threshold sometimes cannot be found. In order to find the optimal threshold with a low computational cost, Tsou et al. [22] proposed an efficient optimal pixel grouping scheme for AMBTC in 2008. For the purpose of improving the efficiency of searching the optimal threshold, we propose a new pixel grouping scheme for AMBTC based on PSO. Meanwhile, by using PSO algorithm, the speed of searching the optimal threshold can be flexibly changed according to the two coefficients, the population of initialized particles and the maximum iteration time of PSO. The experimental results show that our scheme is efficient to find a near optimal threshold when the block size is large.

The rest of this paper is organized as follows. Section 2 reviews some related work. In Section 3, our scheme is presented. Some experimental results are summarized in Section 4 and our conclusions are drawn in Section 5.

2. **Related work.**

2.1. **AMBTC.**

First, a gray scale image is divided into non-overlapping blocks with size n×n. Then take the average value of this block as the threshold to divide this block into two levels. In order to mark the positions of different levels in a block, a corresponding bitmap that only contains 0 or 1 is generated. If the pixel is larger than the average value, this pixel belongs to the high level group and the corresponding bit in the bitmap is set to 1. Otherwise, the pixel belongs to the low level group and the corresponding bit in the bitmap is set to 0. After a bitmap is generated for this block, two mean values of the two groups are computed and are taken as two quantization values. At the decoder, the two quantization values combined with its corresponding bitmap are used to reconstruct the original image block.

The mean value u of each block is computed as follows:

$$u = \frac{1}{n \times n} \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}. \tag{1}$$

where x_{ij} represents the pixels in the block and n×n represents the block size. The two quantization values, the high level h and the low level l are calculated according to Eqs. (2) and (3).

$$h = \frac{1}{n_h} \sum_{x_{ij}>u} x_{ij}, i, j \in 1, 2, ..., n, \tag{2}$$

$$l = \frac{1}{n_l} \sum_{x_{ij} \leq u} x_{ij}, i, j \in 1, 2, ..., n, \tag{3}$$

where n_h and n_l are two numbers of the pixels belonging to the high level group and the low level group, respectively. The bitmap B=$\{b_{ij}|b_{ij} \in \{0, 1\}, i, j \in \{1, 2, ..., n\}\}$ is generated according to the following equation.

$$b_{ij} = \begin{cases} 1, \text{ if } x_{ij} > u \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

An example of AMBTC encoding and decoding procedures are shown in FIGURE 1. FIGURE 1 (a) is an original image block with size 4×4. The mean value of this block is 57 according to Eq. (1). Then the mean value is taken as the threshold to divide this block into two levels. If the pixel is greater than 57, the corresponding bit in the bitmap is set to 1. Otherwise, the corresponding bit is set to 0. The bitmap is shown in FIGURE 1 (b) and the two quantization values are calculated according to Eqs. (2) and (3), respectively. Here, we can get h=68 and l=46. Finally, h, l and the corresponding bitmap B can be used to recover the original image block that is shown in FIGURE 1 (c).

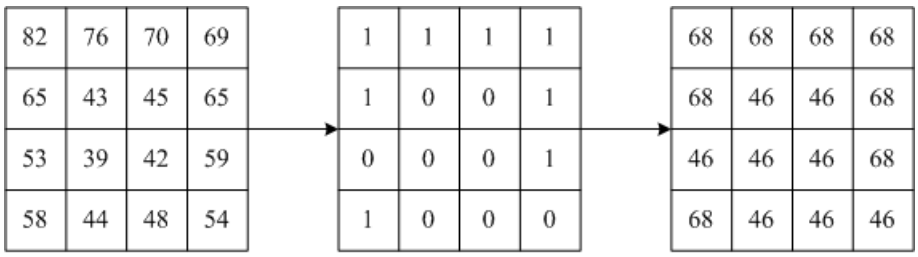

FIGURE 1. An example of AMBTC encoding and decoding: (a) original image block; (b) bitmap B; (c) reconstructed image block

2.2. **Tsou et al.s scheme.** A grayscale image is first divided into a number of non-overlapping blocks of k pixels. A sorted sequence is generated by sorting these k pixels in ascending order. If all the pixels of this block are equal, place these pixels into one group. Otherwise, all pixels are divided into two groups. If the pixel value is smaller than the threshold value, place it into group one. Otherwise, put it into group two. For each block, there are k-1 possible ways to divide k pixels into two groups. In order to find the best threshold for each block, all the possible grouping ways are tried.

A sorted sequence of all pixels in a block is defined as S=(p_1,p_2,...,p_k), where $p_1 \leq p_2 \leq ... \leq p_k$. Here the r^{th} pixel p_r in the sequence is regarded as the threshold to divide

the pixels into two groups. And the squared Euclidean distance $SED(p_r)$ which can record the image distortion between the original image block and the compressed block is calculated according to the corresponding threshold by Eq. (5).

$$SED(p_r) = \sum_{i=1}^{r}[p_i - m_{(p_1,p_r)}]^2 + \sum_{i=r+1}^{k}[p_i - m_{(p_{r+1},p_k)}]^2, \tag{5}$$

where $m_{(p_1,p_r)}$ and $m_{(p_{r+1},p_k)}$ are two mean values of pixels ranged from 1 to r and r to k, respectively. The two mean values are computed as in the following equations.

$$\begin{cases} m_{(p_1,p_r)} = \frac{1}{r}\sum_{i=1}^{r} p_i \\ m_{(p_{r+1},p_k)} = \frac{1}{k-r+1}\sum_{i=r+1}^{k} p_i \end{cases} \tag{6}$$

In order to reduce the encoding time in finding the optimal threshold, a possible way is to explore the relationship from $SED(p_r)$ to $SED(p_{r+1})$. Instead of recalculate the two mean values, the two new mean values can be computed according to its previous mean values as in the following equations.

$$\begin{cases} m_{(p_1,p_{r+1})} = m_{(p_1,p_r)} + \frac{p_{r+1} - m_{(p_1,p_r)}}{r+1} \\ m_{(p_{r+2},p_k)} = m_{(p_{r+1},p_k)} + \frac{m_{(p_{r+1},p_k)} - p_{r+1}}{k-r-1} \end{cases} \tag{7}$$

Based on Eq (7), the new squared Euclidean distance $SED_{p_{r+1}}$ can be calculated as Eq (8).

$$SED(p_{r+1}) = SED(p_r) + \frac{r[p_{r+1} - m_{(p_1,p_r)}]^2}{r+1} - \frac{(k-r)[m_{(p_{r+1},p_k)} - p_{r+1}]^2}{k-r-1}, \tag{8}$$

The basic steps of this algorithm are summarized as the following.

Step 1. Generate a sequence of all pixels in a block in ascending order.

Step 2. Compute $SED(p_1)$ according to Eq. (5) and set r=1.

Step 3. If $r < k$+1, calculate $SED(p_r)$ according to Eq. (8). Otherwise, go to Step 4.

Step 4. Find the minimal squared Euclidean distance value $SED(p_i)$, where $SED(p_1)$=$\{\min[SED(p_1),\ SED(p_2), \ldots, SED(p_r)$ so that the best threshold is p_i.

Step 5. The best threshold p_i is used to divide the pixels into two groups with the least distortion between the original image block and the compressed image block.

2.3. **Particle swarm optimization.** Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart in 1995 [23]. It is a kind of bionic algorithm that is based on the social behavior of flocking birds and schooling fish when searching for food. In PSO, individual members of its population can profit from the discoveries and previous experience of all other members during the searching for food. And the social sharing of information among conspecifics offers an evolutionary advantage. Recently, PSO has attracted great attention in the fields of evolutionary computing, optimization and many others [24, 25, 26, 27].

PSO is initialized with a population of random solutions. Each solution is called a particle that initializes random position and the corresponding velocity. The fitness values of all the initialized particles are computed and used to select the particle that has the best fitness value as the global best solution g_{best}. Meanwhile, each particle is defined as local best solution l_{best}. In each iteration, the position of each particle is updated according to its velocity. The new velocity of each particle is updated as follows [28]:

$$v_i(t+1) = wv_i(t) + c_1r_1(l_{best} - x_i(t)) + c_2r_2(g_{best} - x_i(t)), \tag{9}$$

where $v_i(t)$ and $v_i(t+1)$ represent current velocity and updated velocity of particle i, w) is inertia factor, c_1 and c_2 are two acceleration coefficients, and r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0,1]. The parameter v_{max} is used to limit the velocity of each particle as shown in Eq. (10). If the velocity is too large, particle may fly past good solutions. The position of each particle is updated according to the following equation.

$$v_i(t+1) = \begin{cases} v_{max}, \text{ if } v_i(t+1) > v_{max}, \\ v_i(t+1), \text{ otherwise.} \end{cases} \tag{10}$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{11}$$

where $x_i(t)$ and $x_i(t+1)$ represent the current position and the updated position of particle i, respectively.

The entire process of standard PSO is summarized as the following steps:

Step 1. Initialize a population of particles with random positions and velocities.

Step 2. Define each particle as its local best particle l_{best}. Calculate the fitness value of all the initialized particles, then find the particle that has the best fitness value and define this particle as the global best particle g_{best}.

Step 3. Update the velocity and position of each particle according to Eq. (9) to Eq. (11).

Step 4. Compute the fitness values of the updated particles.

Step 5. For each particle, compare its new fitness value with that of l_{best}. If the new fitness value is better, the local best particle will be replaced by this new particle. Otherwise, the local best particle will not be changed.

Step 6. For each particle, compare its new fitness value with that of g_{best}. If the new fitness value is better, the global best particle will be replaced by this new particle. Otherwise, the global best particle will not be changed.

Step 7. If the terminal condition is met, output the global best particle and its corresponding fitness value. Otherwise, go back to Step 3.

3. **Proposed scheme.** In this paper, we propose a novel optimal pixel grouping scheme for AMBTC based on PSO. The main idea of our scheme is to find a near optimal threshold for each block in order to minimize the distortion of the whole image. PSO is applied to search the optimal threshold for each block compressed by AMBTC. Illustration of searching the optimal threshold for each block based on PSO is shown in FIGURE 2. The details are described as follows.

Input: An original image block

Output: the near optimal threshold for this block

Step 1. Initialize a population of particles.

First, a gray scale image is divided into some non-overlapping blocks with size $n \times n$. For each block, find the minimum value min and the maximum value max. Initialize a population of particles with random values between min and max that is shown in Eq. (12).

$$p_i(t) = rand(max - min) + min, 1 \leq i \leq k, \tag{12}$$

where $rand$ represents a random number distributed in the range of [0,1], $p_i(t)$ represents the value of i^{th} particle, and t represents the initial state. For each particle, the corresponding velocity is initialized as the following equation:

$$v_i(t) = rand(max - min), i = 1, 2, ..., k. \tag{13}$$

Step 2. Determine the local best particle l_{best} and global best particle g_{best}.

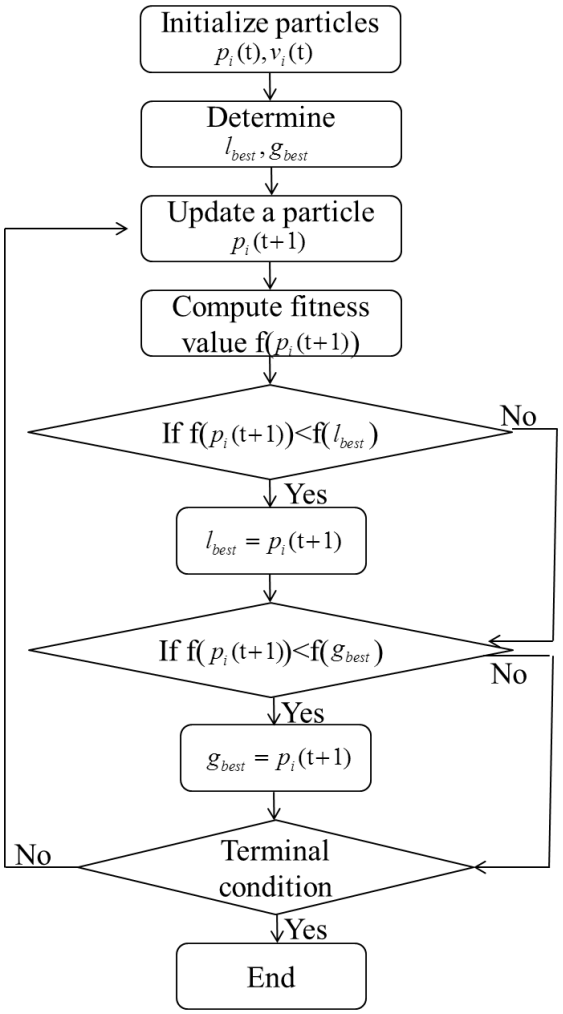

FIGURE 2. The algorithm of searching optimal threshold for each block based on PSO

Each particle is denoted as its local best particle l_{best}. The fitness function is defined as the mean squared error between the original image block x_{ij} and the recovered image block x'_{ij} as in Eq. (14), and each particle is used as the threshold to reconstruct the original image block.

$$f(p_i(t)) = \frac{1}{n \times n} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_{ij} - x'_{ij})^2. \tag{14}$$

where x'_{ij} is reconstructed by the following equation.

$$x'_{ij} = \begin{cases} h \text{ if } x_{ij} > p_i(t), \\ l \text{ otherwise.} \end{cases} \tag{15}$$

where h and l are two quantization values of the reconstructed image block which are defined in Eqs. (16) and (17), respectively. If the original pixel is greater than $p_i(t)$, this pixel belongs to high level group. Otherwise, the pixel belongs to the low level group. And n_h and n_l are two numbers of the pixels belonging to the high level group and the

low level group, respectively.

$$h = \frac{1}{n_h} \sum_{x_{ij} > p_i(t)} x_{ij}, \tag{16}$$

$$l = \frac{1}{n_l} \sum_{x_{ij} \leq p_i(t)} x_{ij}. \tag{17}$$

After we calculate the fitness values of all the initialized particles, find the particle who has the minimum fitness value and denote it as the global best particle g_{best}.

Step 3. Update particles according to its updated velocity.

The velocity of each particle is updated as the following equation.

$$v_i(t+1) = wv_i(t) + c_1 r_1 (l_{best} - p_i(t)) + c_2 r_2 (g_{best} - p_i(t)), \tag{18}$$

where w is set to 0.4, c_1 and c_2 are both set to 1. Then the value of each particle is updated according to its updated velocity as Eq (19).

$$p_i(t+1) = p_i(t) + v_i(t+1), \tag{19}$$

Instead of restricting the maximum velocity of each particle, the value of each particle should be restricted among the values of pixels in this block that is shown in Eq. (20). If the updated value is greater than the maximum value max, the value is replaced by max. If the updated value is less than the minimum value min, the value is substituted by min. Otherwise, the updated value is not changed.

$$p_i(t+1) = \begin{cases} max, \text{ if } p_i(t) > max, \\ min, \text{ if } p_i(t) < min, \\ p_i(t+1), \text{ otherwise.} \end{cases} \tag{20}$$

Step 4. Calculate the fitness value of the updated particle.

The updated particle is used as the threshold to divide the block into two groups. Meanwhile, a bitmap is generated according to the pixel belonging to different groups. Compute the two mean values of all pixels belonging to high level group and low level group, respectively. The image block can be recovered by the two mean values and the bitmap. Then the fitness value is calculated by using Eq. (14).

Step 5. Update local best particle l_{best}.

Compare the new fitness value $f(p_i(t+1))$ with $f(l_{best})$. If $f(p_i(t+1)) < f(l_{best})$, the local best particle l_{best} is updated as $p_i(t+1)$, as well as its fitness value. Otherwise, l_{best} is not updated.

Step 6. Update global best particle g_{best}.

Compare the new fitness value $f(p_i(t+1))$ with $f(g_{best})$. If $f(p_i(t+1)) < f(g_{best})$, the global best particle g_{best} is updated as $p_i(t+1)$, as well as its fitness value. Otherwise, g_{best} is not updated.

Step 7. Termination.

If the iteration time achieves the maximum value m, output the near optimal threshold which is the global best particle g_{best}. Otherwise, go back to Step 3.

Finally, the near optimal threshold is used to perform BTC compression. By using this value, the block is divided into two groups. The mean values of the two groups are computed and are taken as two quantization values of this block. The bitmap is generated according to the pixels belonging to different groups. And the two quantization values and the corresponding bitmap are preserved to restore the original image block.

4. **Experimental results.** In this paper, we proposed a novel pixel grouping scheme for AMBTC based on PSO. Some experimental results are shown to prove the feasibility of our scheme in this section. The simulation environment of our experiments was a PC with a 3.4 GHz CPU and 8 GB RAM. The programming language was Matlab. FIGURE 6 showed the six grayscale images that were used in our scheme. From TABLE 1 to TABLE 6, the population of particles is set to 4 and the maximum iteration time is set to 8 in the proposed scheme.

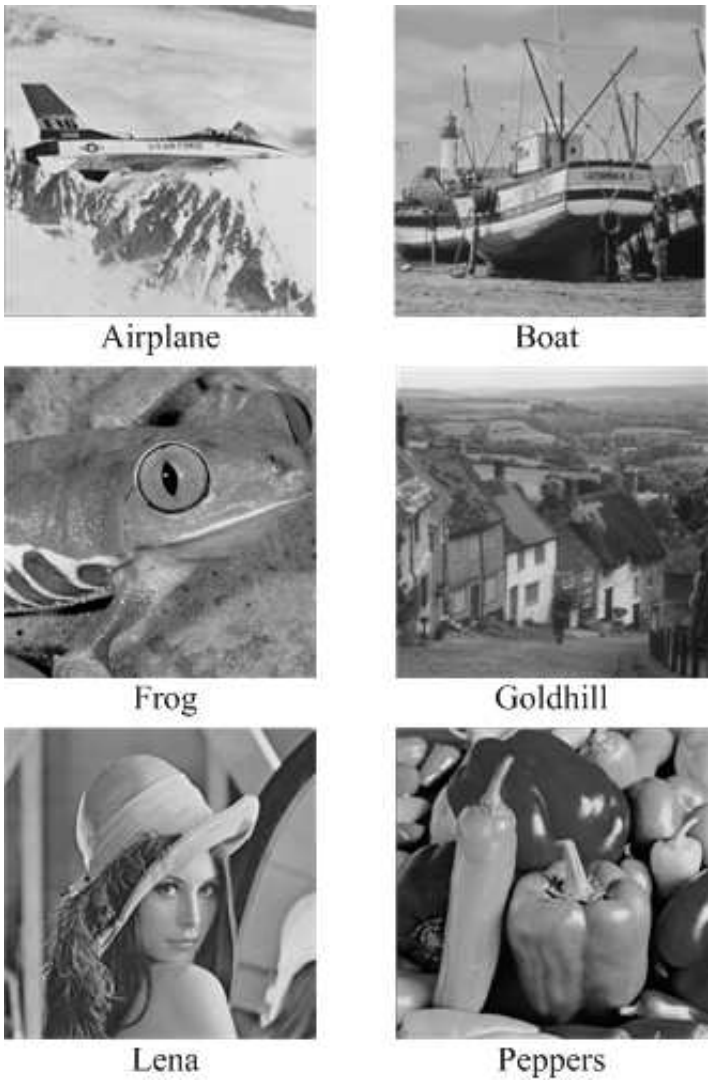

FIGURE 3. Six grayscale images

To measure the visual quality between the reconstructed image and the original image, the peak signal to noise ratio ($PSNR$) was used. The higher $PSNR$ value means the better image quality. It is defined as follows:

$$PSNR = 10log_{10}(\frac{255^2}{MSE}), \tag{21}$$

where MSE denotes mean squared error (MSE) between the original image O_{ij} and the recovered image R_{ij} of $M \times N$ pixels as in Eq. (22).

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (O_{ij} - R_{ij})^2. \tag{22}$$

TABLE 1, TABLE 2, and TABLE 3 show the PSNR comparison of different schemes when block sizes are set to 4×4, 8×8, and 16×16, respectively. From the experimental results, AMBTC has the lowest PSNR values while OBTC and Tsou et al.s scheme achieve the best image quality, and the proposed scheme achieves a near optimal image quality. As the block size increases, the image quality decreases. The average image quality of our scheme decreases only 0.14 dB, 0.16 dB, and 0.17 dB compared with OBTC and Tsou et al.s scheme when the block size is set to 4×4, 8×8, and 16×16, respectively. However, the average improvement of our scheme is 0.45 dB, 0.50 dB, and 0.65 dB compared with AMBTC when the block size is set to 4×4, 8×8, and 16×16, respectively.

TABLE 1. PSNR comparison of different schemes when the block size is set to 4×4 (dB)

Images	**AMBTC [16]**	**OBTC [20]/Tsou et al.'s scheme [22]**	**Proposed scheme**
Airplane	31.98	32.75	32.62
Boat	31.55	32.11	31.97
Frog	28.63	29.04	28.89
Goldhill	32.87	33.43	33.28
Lena	33.23	33.78	33.64
Peppers	33.21	33.86	33.73
Average	31.91	32.50	32.36

TABLE 2. PSNR comparison of different schemes when the block size is set to 8×8 (dB)

Images	**AMBTC [16]**	**OBTC [20]/Tsou et al.'s scheme [22]**	**Proposed scheme**
Airplane	28.85	29.74	29.59
Boat	28.60	29.22	29.05
Frog	26.52	26.88	26.72
Goldhill	29.93	30.47	30.32
Lena	29.93	30.58	30.43
Peppers	29.46	30.35	30.19
Average	28.88	29.54	29.38

TABLE 4, TABLE 5, and TABLE 6 show the encoding time of different schemes when block sizes are set to 4×4, 8×8, and 16×16, respectively. When the block size is set to 4×4, the proposed scheme costs the most time than other schemes as shown in TABLE 4. However, when block size increases, our scheme is more efficient than OBTC and Tsou et al.s scheme. The average encoding time of our scheme is 2.24 s and 1.09 s when the block sizes are set to 8×8 and 16×16, respectively, which is lower than the average encoding time of OBTC and Tsou et al.s scheme.

Based on the above experimental results, AMBTC costs least time while maintaining the worst image quality with different block sizes among these schemes. Both OBTC and Tsou et al.s scheme achieve the best image quality, but Tsou et al.s scheme is more efficient. As the block size increases, the encoding time of both OBTC and Tsou et al.s scheme increases while the encoding time of our scheme decreases. Although the proposed

TABLE 3. PSNR comparison of different schemes when the block size is set to 16×16 (dB)

Images	AMBTC [16]	OBTC [20]/Tsou et al.'s scheme [22]	Proposed scheme
Airplane	26.64	27.72	27.52
Boat	26.42	27.20	27.05
Frog	24.94	25.36	25.21
Goldhill	27.82	28.36	30.32
Lena	27.19	28.06	27.90
Peppers	26.07	27.26	27.12
Average	26.51	27.33	27.16

TABLE 4. Encoding time of different schemes when the block size is set to 4×4 (s)

Images	AMBTC [16]	OBTC [20]	Tsous scheme [22]	Proposed scheme
Airplane	0.74	4.65	1.86	7.77
Boat	0.72	4.46	1.84	7.79
Frog	0.72	4.70	1.82	7.75
Goldhill	0.73	4.76	1.82	7.87
Lena	0.72	4.49	1.84	7.88
Peppers	0.73	4.49	1.84	7.75
Average	0.73	4.63	1.84	7.80

TABLE 5. Encoding time of different schemes when the block size is set to 8×8 (s)

Images	AMBTC [16]	OBTC [20]	Tsous scheme [22]	Proposed scheme
Airplane	0.74	4.65	1.86	7.77
Boat	0.72	4.46	1.84	7.79
Frog	0.72	4.70	1.82	7.75
Goldhill	0.73	4.76	1.82	7.87
Lena	0.72	4.49	1.84	7.88
Peppers	0.73	4.49	1.84	7.75
Average	0.73	4.63	1.84	7.80

scheme achieves the near optimal image quality, it requires the lower computational cost than OBTC and Tsou et al.s scheme when the block sizes are set to 8×8 and 16×16.

In addition, our scheme is flexible because the population of initialized particles k and the maximum iteration time m can be changed. This flexibility can provide flexible tradeoff between the encoding time and recovered quality required. TABLE 7 shows the PSNR and encoding time comparison of different population of initialized particles when the block size is set to 8×8. And TABLE 8 shows the PSNR and encoding time comparison of different maximum iteration time when the block size is set to 8×8. From the experimental results, either the population of initialized particles or the maximum

TABLE 6. Encoding time of different schemes when the block size is set to 16×16 (s)

Images	**AMBTC [16]**	**OBTC [20]**	**Tsous scheme [22]**	**Proposed scheme**
Airplane	0.33	5.52	2.61	2.28
Boat	0.29	5.37	2.60	2.22
Frog	0.31	5.57	2.59	2.21
Goldhill	0.29	5.54	2.64	2.27
Lena	0.32	5.21	2.61	2.23
Peppers	0.31	5.55	2.62	2.22
Average	0.31	5.46	2.61	2.24

iteration time increases, the image quality will be improved, as well as the encoding time.

TABLE 7. PSNR and encoding time comparison of different population of initialized particles when the block size is set to 8×8

Images	**k=2,m=4**		**k=4,m=4**		**k=8,m=4**	
	PSNR(dB)	Time(s)	PSNR(dB)	Time(s)	PSNR(dB)	Time(s)
Airplane	28.45	1.38	29.59	2.28	29.62	3.72
Boat	28.00	1.39	29.05	2.22	29.09	3.58
Frog	25.70	1.36	26.72	2.21	26.74	3.63
Goldhill	29.10	1.37	30.32	2.27	30.37	3.65
Lena	29.13	1.34	30.43	2.23	30.46	3.62
Peppers	28.95	1.39	30.19	2.22	30.22	3.63
Average	28.22	1.37	29.38	2.24	29.42	3.46

TABLE 8. PSNR and encoding time comparison of different maximum iteration time when the block size is set to 8×8

Images	**k=4,m=2**		**k=4,m=4**		**k=4,m=8**	
	PSNR(dB)	Time(s)	PSNR(dB)	Time(s)	PSNR(dB)	Time(s)
Airplane	29.16	1.74	29.59	2.28	29.62	3.35
Boat	28.59	1.63	29.05	2.22	29.10	3.34
Frog	26.40	1.54	26.72	2.21	26.75	3.30
Goldhill	29.81	1.52	30.32	2.27	30.36	3.34
Lena	29.86	1.50	30.43	2.23	30.47	3.32
Peppers	29.78	1.52	30.19	2.22	30.24	3.29
Average	28.93	1.58	29.38	2.24	29.42	3.32

5. **Conclusions.** In this paper, we propose a novel pixel grouping scheme for AMBTC based on PSO. First, the image is divided into many non-overlapping blocks. Second, we adopt PSO to search the optimal threshold for pixel grouping to minimize the distortion of compressed images. By using PSO algorithm, the speed of searching the optimal threshold can be flexibly changed according to two coefficients, the population of initialized particles and the maximum iteration time. When the near optimal threshold is found, each block is

divided into two groups depending on whether the pixel value is larger than the threshold. The mean values of two groups are taken as two quantization levels of the block and the corresponding bitmap is generated according to the pixels belonging to different groups. Finally, the two quantization values and the corresponding bitmap are used to recover the original image block. The experimental results have shown that the proposed scheme achieves a near optimal image quality with a low computational cost when the block sizes are set to 88 and 1616, respectively. Moreover, our scheme can provide flexible tradeoff between the encoding time and recovered quality required. In the further, our scheme is suiTABLE for the practical multimedia applications for its efficiency and good visual quality.

Acknowledgment. This work has been supported in part by National Natural Science Foundation of China (No. 61272262), Program for New Century Excellent Talent in Universities (NCET-12-1037), International Cooperative Program of Shanxi Province (No. 2015081015), and Scientific and Technological project of Shanxi Province (2015031003-2).

REFERENCES

[1] X. Wu, and N. Memon, Context-based, adaptive, lossless image coding, *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437-444, 1997.

[2] N.V. Boulgouris, D. Tzovaras, and M.G. Strintzis, Lossless image compression based on optimal prediction, adaptive lifting, and conditional arithmetic coding, *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 1-14, 2001.

[3] R.A. DeVore, B. Jwwerth, and B.J. Lucier, Image compression through wavelet transform coding, *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 719-746, 1992.

[4] K. Belloulata, and J. Konrad, Fractal image compression with region-based functionality, *Transactions on Image Processing*, vol. 11, no. 4, pp. 351-362, 2002.

[5] R.M. Gray, Vector quantization, *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4-29, 1984.

[6] Y.C. Hu, Predictive moment preserving block truncation coding for graylevel image compression, *Journal of Electronic Imaging*, vol. 13, no. 4, pp. 871-877, 2014.

[7] E.J. Delp, and O.R. Mitchell, Image compression using block truncation coding, *IEEE Transactions on Communications*, vol. 27, no. 9, pp. 1335-1342, 1979.

[8] I.C. Chang, Y.C. Hu, W.L. Chen, and C.C. Lo, High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding, *Signal Processing*, vol. 108, pp. 376-388, 2015.

[9] P.Y. Tsai, C.C. Chang, and Y.C. Hu, An adaptive two-stage edge detection scheme for digital color images, *Real-Time Imaging*, vol. 8, no. 4, pp. 329-343, 2002.

[10] J.M. Guo, H. Prasetyo, and J.H. Chen, Content-Based Image Retrieval Using Error Diffusion Block Truncation Coding Features, *IEEE Transactions on Circuits Systems for Video Technology*, vol. 25, no. 3, pp. 466-481, 2014.

[11] X. Wang, and Z. Wang, The method for image retrieval based on multi-factors correlation utilizing block truncation coding, *Pattern Recognition*, vol. 47, no. 10, pp. 3293-3303, 2014.

[12] D. Ou, and W. Sun, Reversible AMBTC-based secret sharing scheme with abilities of two decryptions, *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 1222-1239, 2014.

[13] D. Ou, L. Ye, and W. Sun, User-friendly secret image sharing scheme with verification ability based on block truncation coding and error diffusion, *Journal of Visual Communication and Image Representation*, vol. 29, pp. 46-60, 2015.

[14] D. Halverson, N.C. Griswold, and G.L. Wise, A generalized block truncation coding algorithm for image compression, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, no. 3, pp. 664-668, 1984.

[15] V. Udpikar and J. P. Raina, Modified algorithm for block truncation coding of monochrome images, *Electronics Letters*, vol. 21, no. 20, pp. 900-902, 1985.

[16] M.D. Lema, O.R. Mitchell, Absolute moment block truncation coding and applications to color images, *IEEE Transactions on Communications*, vol. 32, no. 10, pp. 1148-1157, 1984.

[17] L. Hui, An adaptive block truncation coding algorithm for image compression, *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2233-2236, 1990.
[18] N. Efrati, H. Licztin, and H.B. Mitchell, Classified block truncation coding-vector quantization: an edge sensitive image compression algorithm, *Signal Processing: Image Communication*, vol. 3, pp. 275-283, 1991.
[19] I. Mor, Y. Swissa, and H.B. Mitchell, A fast nearly optimum equispaced 3-level block truncation coding, *Signal Processing: Image Communication*, vol. 6, no. 5, pp. 397-404, 1994.
[20] M. Kamel, C. T. Sun, and G. Lian, Image compression by variable block truncation coding with optimal threshold, *IEEE Transactions on Signal Processing*, vol. 39, no. 1, pp. 208-212, 1991.
[21] C.Y. Yang, and J.C. Lin, EBTC: an economical method for searching the threshold of BTC compression, *Electronics Letters*, vol. 32, no. 20, pp. 1870-1871, 1996.
[22] C.C. Tsou, Y.C. Hu, and C.C. Chang, Efficient optimal pixel grouping schemes for AMBTC, *Imaging Science Journal*, vol. 56, no. 4, pp. 217-231, 2008.
[23] J. Kennedy, and R.C. Eberhart, Particle swarm optimization, *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1942-1948, 1995.
[24] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu and Q.X. Wang, Particle swarm optimization-based algorithms for TSP and generalized TSP, *Information Processing Letters*, vol. 103, no. 5, pp. 169-176, 2007.
[25] J.F. Chang, S.C. Chu, J.F. Roddick and J.S. Pan, A Parallel Particle Swarm Optimization Algorithm with Communication Strategies, *Journal of Information Science and Engineering*, vol. 21, no. 4, pp. 809-818, 2005.
[26] P.Y. Hsu, and Y.L. Yeh, Study on Flood Para-Tank Model Parameters with Particle Swarm Optimization, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 6, no. 5, pp. 911-923, 2015.
[27] M. Zhao, J.S. Pan, and S.T. Chen, Optimal SNR of Audio Watermarking by Wavelet and Compact PSO Methods, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 6, no. 5, pp. 833-846, 2015.
[28] B. Liu, L. Wang, and Y.H. Jin, An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling, *IEEE Transactions on Systems, Man, Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 18-27, 2007.