Proceedings of the 6th WSEAS International Conference on Multimedia Systems & Signal Processing, Hangzhou, China, April 16-18, 2006 (pp125-131)

A Color Re-Indexing Scheme Using Genetic Algorithm

MING-NI WU[2], CHIA-CHEN LIN[3] and CHIN-CHEN CHANG[12]
[1]Department of Information Engineering and Computer Science
Feng Chia University
Taichung, Taiwan, 40724, R.O.C
[2]Department of Computer Science and Information Engineering
National Chung Cheng University
Chaiyi, Taiwan 621, R. O. C.

[3]Department of Computer Science and Information Management
Providence University
Taichung, Taiwan 433, R. O. C.

Abstract: - An indexed image consists of the lookup color table and index sequence. To further reduce the transmission size of indexed images, many lossless image compression techniques are applied. Some of them perform the re-indexing of color indices according to perceptive similarity between different colors. Recording the color table, however, can lead to lower entropy of predictive errors. Several schemes focus on color re-indexing have been proposed. However, it is a time-consuming process to find out an optimal order of a color table. Therefore, most of them only can provide good performance on either execution time or compression ratio. In this paper, we explore a heuristic method based on genetic algorithm to improve the performances on execution time and compression ratio and keep the balance between them. Experimental results further confirm that our proposed scheme only takes one-third the execution time to provide compression ratios those are very close to Memon et al.'s scheme's. The effectiveness of our proposed scheme is acceptable and proved. Moreover, our scheme is particularly suited for real-time applications that require higher compression ratio.

Key-Words: Re-indexing, Palette-based image, Genetic algorithm

1 Introduction

There are various image compression schemes for different kinds of image characteristics. For an image that comprises a small amount of colors for a large color space, the palette-based (or color-mapped based) compression method often provides good efficiency. This kind of compressed image consists of two different parts. One is the fixed lookup color table. That is the color collection for the image. Another is the index sequence of pixels to indicate the color's position in the lookup color table. Many lossless compression algorithms adopt a differential-predictive approach to decode the index sequence. Most prediction schemes assume that neighboring pixels have similar intensity. We can infer that if the index sequence is smoother then lower predictive error can be obtained. On the contrary, the higher predictive error may occur while the index sequence is variable. To maintain a smooth index sequence, and to make sure the higher compression ratio and lower predictive errors, the color re-indexing problem is raised.

The color re-indexing concept is to reorder the lookup color table. Different index sequences are generated by using the color re-indexing procedure. Difference index sequences imply different predictive errors. To reduce the predictive errors, the variableness of index sequences must been reduced in advance. Therefore, the color re-indexing problem also can be treated as a smoothness maximization problem [4].

Many related researches have been proposed in different literatures. These researches can be classified into two categories. The first one performs the re-indexing of color indices according to perceptive similarity between different colors. In [6], Zeng et al. proposed a greed look-ahead scheme to optimize the assignment of index values to colors. They tried to assign color index values to symbols those are frequently located next to each other to improve the compressing ratio. In [1], Battiato et al. translated the color re-indexing problem to traveling salesman problem (TSP). They constructed a weighted graph according to the adjacent frequencies. Then, a heuristic algorithm was applied to select the re-indexing associated with the heaviest Hamiltonian path. The other one is to treat color re-indexing problem as an entropy minimization problem. Memon et al. [4] investigated the problem to minimize the absolute sum of prediction errors. They proposed a simulated annealing to search for a locally optimal ordering to archive a local optimal solution. To reduce the large convergence time for simulated annealing function, they also proposed a pair wise merging method. It is a heuristic selection function, which gives a result very close to those obtained by the much more computationally expensive simulated annealing approach. In general, Zeng et al.'s and Battiato et. al's methods have lower execution time but with worse compression ratios. On the contrary, Memon et al.'s method takes longer execution time but gets better compression ratios.

In this paper, we try to adopt another approach, the genetic algorithm, to deal with the color re-indexing problem. Genetic algorithm is a randomized search approach that is generally used in various fields to solve the optimization problems. It repeats a generation procedure to obtain a near optimum solution. Based on GA's characteristic, our proposed scheme can keep the balance between the compression ratio and execution time.

In the next section, we introduce some relative works. In Section 3, the proposed scheme will be described. Section 4 shall demonstrate the experimental results and discussions. Finally, the conclusion is given in Section 5.

2 Relative Works

In this section, the color re-indexing problem will be pointed out first. Next, we shall introduce two related re-indexing methods: one is Memon et al.'s pairwise merge method and the other is Battiato et al.'s TSP-based scheme. At last, the genetic algorithm (GA) that will be used in our scheme to find out the near optimal order of re-indexing colors will be introduced.

2.1 The Color Re-indexing Problem

Many commercial image processing and geographic information systems adopt the color mapping system to represent color images and save storage space at the same time. In the color mapping system, a fixed lookup table is generated in advance to record the relationship between the colors and indices. Then indexed images (also called index sequences) encode colors using the fixed lookup color table. Each entry in the color table is generally a triple of RGB values. For each pixel in an image, only the indices of corresponding colors need to be stored. Fig. 1(a) shows an example of a simple image with pixels. The entry in each square is a triple of RGB values for the corresponding pixel. To represent the encoded result of color mapping system, a lookup table I and an index sequence I for the example image (shown in Fig. 1(a)) are presented in Fig. 1(b). In Fig. 1(a), the first entry of RGB values is (100,20,50); therefore, the first index value of index sequence I is "0" by looking up the lookup color table I.

(100,20,5	(60,150,200	(60,150,200	(140,140,12
(100,20,5	(30,70,80)	(30,70,80)	(60,150,200
(140,140,	(100,20,50)	(60,150,200	(100,20,50)
(100,20,5	(60,150,200	(140,140,12	(100,20,50)

(a) An example image

Lookup Color Table I	Lookup Color Table II
0: (100,20,50)	0: (140,140,120)
1: (60,150,200)	1: (100,20,50)
2: (140,140,120)	2: (30,70,80)
3: (30,70,80)	3: (60,150,200):

Index Sequence I

0	1	1	2
1	3	3	1
2	0	1	0
0	1	2	0

(b) A lookup table I and an index sequence I

Index Sequence II

1	3	3	0
1	2	2	3
0	1	3	1
1	3	0	1

(c) A lookup table II and an index sequence II

Fig. 1. An example of color mapping system using a lookup color table with different orders

If we reorder the indices of lookup color table I to generate lookup color table II, the corresponding index sequence of original image based on lookup color table II can be generated as index sequence II shown in Fig. 1(c). From Fig.1, we can see that different order of indices in the lookup table produces different index sequences for an image. Different index sequences could lead to different compression efficiency. Therefore, the efficiency of a lossless compression algorithm for indexed images may greatly depend on the assignment of indices in the relative lookup color table. To compress the index sequence, many lossless compression algorithms adopt a differential-predictive approach to encode the index sequence. Smoother index sequence is favorable for compression algorithms. An entropy measure [3] or a difference measure [4] can measure the distribution of an index sequence. We list a simple difference measure as follows.

$$D(S) = \sum_{i=1}^{m\times n-1} |s_i - s_{i-1}| \ , \tag{1}$$

where $m \times n$ is the size of a image and s_i is the ith pixel in the image by raster scan order. In above case, if we apply Equation (1) for Fig. 1(b) and Fig. 1(c), we can get dissimilar values 16 and 22, respectively. It indicates the fact that lossless compression can be optimized by choosing a different platted ordering. Finding an optimal indexing scheme is a crucial step for different lossless compression of indexed images.

2.2 Memon et al.'s Pairwise Merge Method

In 1996, Memon et al. investigated the problem of ordering the color table such that the absolute sum of prediction errors is minimized [4]. They proposed two heuristic solutions for the problem and used them for ordering the color table prior to encoding the image by lossless predictive techniques. The first one used simulated annealing to search for a locally optimal ordering. The second one used pairwise merge heuristic method. The pairwise merge gave results very close to those obtained by the much more computationally expensive simulated annealing approach; therefore, we only introduce Memon et al.'s pairwise merge method in this subsection.

Assume an image has *M* colors. Memon et al.'s pairwise merge method is summarized as follows. summarized as follows.

Step 1: Assign each color number to a different set. There will be M set in the set pool initially. A set will be a particular order of some colors after the following merging procedure.

Step 2: Find the two sets with the lowest perdition error cost.

Setp 2.1 For any two sets a_1, a_2, Λ, a_r (called set 1) and $a'_1, a'_2, \Lambda, a'_s$ (called set 2) in the set pool, we merge them to generate temporary sets according the following rules

$$a_1, a_2, \Lambda, a_r, a'_1, a'_2, \Lambda, a'_s$$
$$a_r, \Lambda, a_2, a_1, a'_1, a'_2, \Lambda, a'_s$$
$$a'_1, a'_2, \Lambda, a'_s, a_1, a_2, \Lambda, a_r$$
$$a'_1, a'_2, \Lambda, a'_s, a_r, \Lambda, a_2, a_1 ,$$

where ai is a color number and set 1 and set 2 has r and s members, respectively.

Setp 2.2 For any temporary set, calculate the cost according to Equation (2):

$$\sum_{i=1}^{k}\sum_{j=1}^{k}|j-i| \bullet t(a_i, a_j) , \qquad (2)$$

where $t(a_i, a_j)$ represents the time that color ai is next to color aj, k is the amount of members in the temporary set.

Setp 2.3 Select a temporary set with the lowest cost, add it to the set pool and delete two original sets that produce the selected set. The amount of sets will be subtracted one for the set pool.

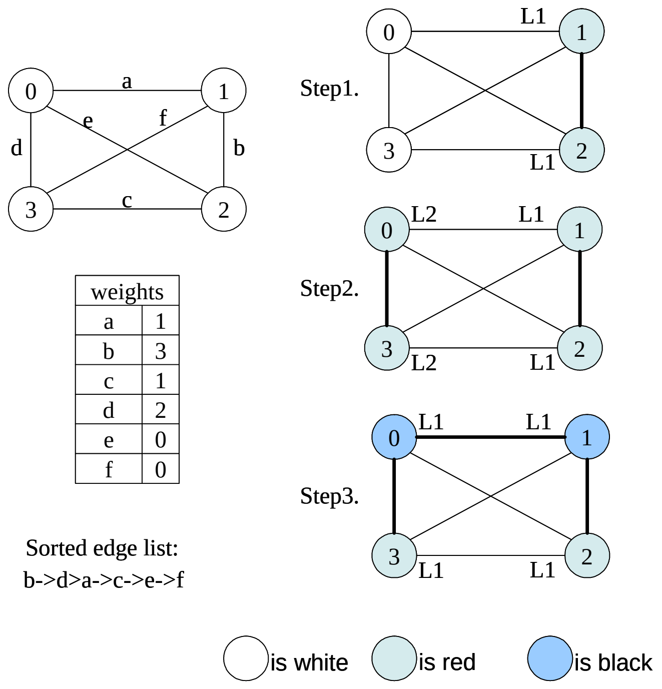
Step 3: Repeat Step 2 until the amount of sets is equal to 1.

Step 4: Output the final set from the set pool to be the order of colors.

2.3 Battiato et al.'s TSP-based Scheme

In 2004, Battiatop et al. restated the re-indexing problem as a graph optimization problem [1]. They believed an optimal re-indexing corresponds to the heaviest Hamiltonian path in a weighted graph. Therefore, they transferred the re-index problem into the TSP problem then tried to found a Hamiltonian path in the weighted graph. Since TSP is an NP-complete problem, they adopted a heuristic method to find an approximate solution.

Assume an image I with M colors. They created an M nodes weighted graph first. Each node represents the color number of the color table. The weight is assigned to represent the occurrence times for the pair of neighboring colors (or called nodes). Next, the edges are sorted in decreasing weight order and all vertices are marked as white in this graph. The algorithm continues incrementally adding to the path non-visited edges with maximal weight. This procedure is detailed in the following steps (shown in Fig. 2).



weights	
a	1
b	3
c	1
d	2
e	0
f	0

Sorted edge list:
b->d>a->c->e->f

Fig. 2 Successive steps of the construction of the Hamiltonian path according to Battiatop et al.'s algorithm

Step 1: Extract the edge e_{ij} of maximal weight that has not been processed.

Step 2: Four cases need to be considered.

Setp 2.1 Both vertices i, j are white. In this case, both of them are set to red and they get the same new label.

Setp 2.2 One of the two vertices is white and the other one is red. In this case, the white vertex becomes red and gets the same label that has been assigned to the red vertex; the red vertex, in turn, becomes black.

Setp 2.3 Both vertices i, j are red. If both vertices have been already visited by the algorithm. Two cases may further explored: 1) If both vertices are the extreme of a chain of vertices with the same label, skip edge e_{ij}. 2) If the vertices are the extreme of two distinct chains of vertices, both of them are set to black and the corresponding labels are unified.

Setp 2.4 One vertex is black. In this case, the edge e_{ij} is skipped.

Step 3: Repeat Steps 1 to 2 until there are unprocessed edges.

The new indexing can be obtained according to the corresponding position of the various color symbols in the Hamiltonian path, scanning the path in one of the two possible directions.

3. The Proposed Scheme

Following the overview of relative works in Section 2, we shall describe the details of the proposed algorithm for re-indexing problem in this section. Let *I* be an image of $m \times n$ pixels and *M* be the amount of distinct colors. There is a color lookup table $C = \{c_0, c_1, \Lambda, c_{M-1}\}$ and an index sequence $S = \{s_0, s_1, \Lambda, s_{m\times n-1}\}$ used to represent the image *I*, when *I* is compressed by a color-mapped algorithm, where c_i is a color value and s_i is a color index of location *i* in lookup table. To further reduce the size of index sequence, a lossless compression technique will be applied. To make sure the efficiency of lossless compression technique can be achieved, we have to minimize the difference of neighbour indices. That means we have to re-arrange the order of indices in the color lookup table.

In this paper, we apply GA to find out the optimal re-indexing order. The evaluation function listed in Equation (1) is our fitness function in our GA algorithm. To prepare the initial chromosome pool, we randomly generate *Q* chromosomes $\{CH_0, CH_1, \Lambda, CH_{Q-1}\}$, where CH_i is one kind order of original lookup color table. For a chromosome $CH_i = h_0 h_1 \Lambda\ h_{M-1}$, h_j indicates the original *j*th order in *C* which is reorder to h_jth order in CH_i. A chromosome pool example for an image with 8 colors is shown in Fig. 3. In this example, the 0th gene of chromosome CH_0 is 4, it indicate the 0th color in *C* is reordered to the 4th color in $CH_{0,}$ the 4th gene of chromosome CH_1 is 5, it indicate the 4th color in *C* is reordered to the 5th color in $CH_{1,}$.

The crossover process is a key step for our algorithm. To speed up the convergence ratio, we adopt a heuristic crossover operation. In this step, two chromosomes CHi and CHj are randomly selected. A random number z is generated to split CHi and CHj into left hand and right hand sides. The right hand sides of CHi and CHj are exchanged. A cross over example is shown in Fig. 4. In this example, we suppose z=4 and two new chromosomes CH_i' and CH_j' are produced. If a new chromosome has any duplicate genes, we only can keep one of them and replace the other one with unused genes. For example, h0=h5=4 and h4=h7=0 are two pair of duplicate genes in CH_i'. To decide which genes will be removed, we calculate the co-occurrence value wi, wi indicate the time of neighbouring color symbols (hi,hi+1). Let assume the co-occurrence value of h0 is 20 (w0) and the average co-occurrence value of h5 is 58 ($w_4 + w_5 \big/ 2$). Since h0 with lower co-occurrence value, h0 will be replaced by an unused gene "5". It is noted that the heuristic replacement scheme will keep high co-occurrence symbol pair and reduce the time of iteration in GA algorithm.

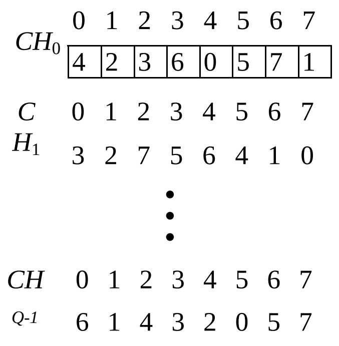


Fig. 3. An example of an initial chromosome pool

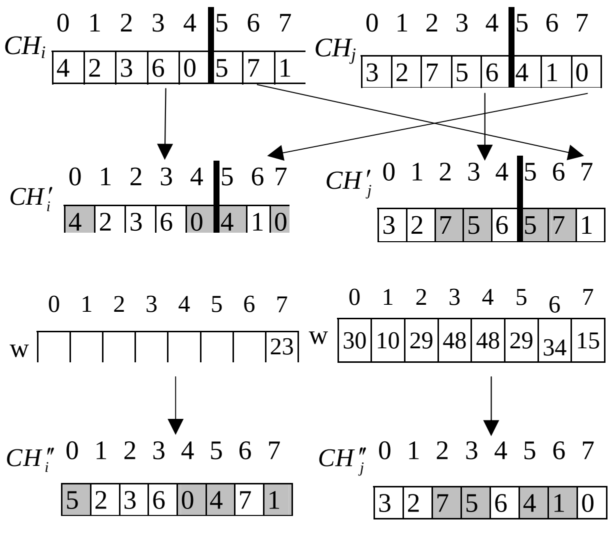


Fig. 4. An example of the heuristic cross over operation

In mutation operation, give a chromosome CH_i, two genes of CH_i are selected randomly and their values are replaced by each other. An example of mutation operation is shown in Fig. 5.

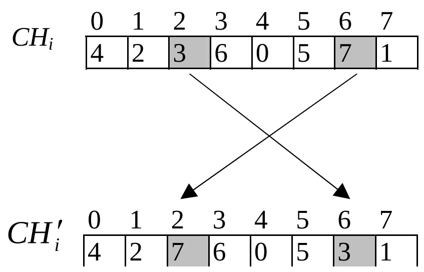


Fig. 5. An example of the mutation operation

Our proposed reordering process is detailed in the following steps.

Step 1: Prepare the initial chromosome pool. Q chromosomes $\{CH_0, CH_1, \Lambda, CH_{Q-1}\}$ are randomly generated. A chromosome CH_i is a particular order of color index sequence and consists of M genes where $CH_i = h_0 h_1 \Lambda\ h_{M-1}$, and $0 \le h_i \le M-1$. In CH_i, h_0 is the new position of color 0 of C, h_1 is the new position of color 1 of C, and so on. For each chromosome, any gene pair (h_i,h_j) has the property $i \neq j$ that implies $h_i \neq h_j$.

Step 2: Evaluation process. For each chromosome CH_l, we can produce $S' = \{s_0', s_1', \Lambda, s_{m\times n-1}'\}$ from S by CH_l mapping where $s_i' = h_{s_i}$. We apply the fitness function (listed in Equation (3)) to calculate the difference sum of S'.

$$F(S') = \sum_{i=0}^{m\times n-1} (s_i' - s_{i-1}') \cdot \qquad (3)$$

If the amount of loops has reached the default threshold, the best chromosome with the lowest $F(S')$ value in the pool will be outputted as the winning chromosome, and the procedure is terminated.

Step 3: Selection process. From all $F(S')$ in the chromosome pool, we preserve the better chromosomes and drop the worse chromosomes.

Step 4: Crossover process. Choose any two chromosomes: $CH_i = p_0 p_1 \Lambda\ p_{M-1}$, $CH_j = q_0 q_1 \Lambda\ q_{M-1}$. A random number z is generated where $0 < z < M-1$. We split CH_i and CH_j into left hand and right hand sides using z, and the right hand sides of CH_i and CH_j are exchanged to get the new offspring as Equation (4).

$$CH_i' = p_0 p_1 \Lambda\ p_{z-1} q_z q_{z+1} \Lambda\ q_{M-1} ,$$
$$CH_j' = q_0 q_1 \Lambda\ q_{z-1} p_z p_{z+1} \Lambda\ p_{M-1} . \qquad (4)$$

This process may produce non-unique genes in CH_i' and CH_j'. To make sure all genes in CH_i' and CH_j' are unique, we can add a post process phase to remove duplicate genes and replacing them by unused genes. These two new offspring CH_i'' and CH_j'' will be added to chromosome pool. Here, we adopt a heuristic replacing method as discussed previously to improve the iteration efficiency.

Step 5: Mutation process. Select one chromosome CH_l from the pool and pick up two random numbers y and z from 0 and M-1. Let $CH_l = h_0 \Lambda\ h_y \Lambda\ h_z \Lambda\ h_{M-1}$. We select the genes h_y and h_z and replace their values with each other. The result is $CH_l' = h_0 \Lambda\ h_z \Lambda\ h_y \Lambda\ h_{M-1}$. I t will be added to the chromosome pool. Go to Step 2.

4. Experimental Results

In this section, we shall present and discuss the experimental results of the proposed scheme. All the programs were written in Borland C++ Builder and were run on a personal computer with the Windows XP operating system. The CPU is Pentium 4 with 512 MB. In our experiments, four test images are processed into many copies of color-mapped images with three different numbers of colors, such as 16, 64 and 256 different colors. Four color-mapped images with 64 colors are shown in Fig. 6.

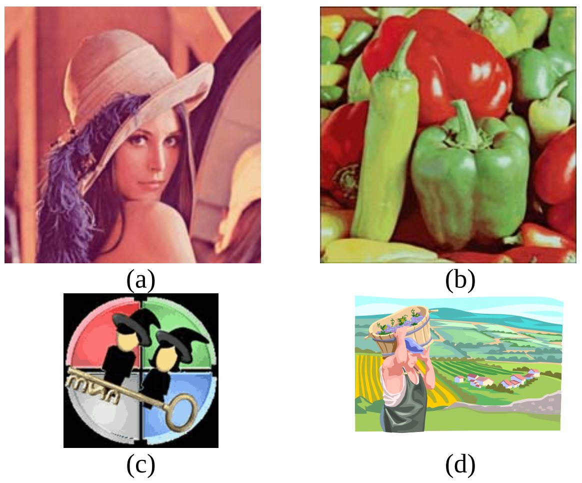


Fig. 6. Four test color-mapped images with 64 different colors: (a) Lena, (b) Pepper, (c) Logo, (d) Farmer

To evaluate the effectiveness of the proposed scheme, we have performed several tests over different sets of indexed images. Using the difference measure presented in Subsection 2.1, the comparisons of difference measure (*D* value) by our fitness function for images with different number of colors are listed in Table 1. The value before "/" is the *D* value of an image with original color map. The value after "/" is the *D* value of an image with re-ordering color mapping after running our algorithm. It is noted that the values of difference measure those with our proposed scheme are decreasing compared with those with original color mapping. This reduction indicates the efficiency of our proposed re-ordering scheme.

Table 1 The variations for difference measure

	Number of Different Colors		
Images	16	64	256
Lena	184153/ 151377	299499/ 235687	386662/ 310365
Peppers	179364/ 131502	263515/ 215632	354103/ 296523
Logo	69487/ 51047	107433/ 85692	137552/ 112542
Farmer	129436/ 99854	213079/ 165488	

In the second experiment, we compare different algorithms' performance. Two previous works including Memon et al.'s and Battiato et al.'s methods were also implemented to get their reordered color mapping. The JPEG2000-lossless algorithm [5] is applied to produce the compression ratio. The results by bit per pixel (bpp) are presented in Table 2. The time comparison is listed in Table 3.

Table 2 The comparisons of compression ratio among three methods (bpp)

Images	Memon et al.'s	Battiato et al.'s	Proposed scheme
Lena (64)	4.35	5.01	4.39
Peppers (64)	4.21	5.23	4.18
Logo (64)	4.06	4.89	4.13
Farmer(64)	3.98	5.04	4.01
	4.15	5.04	4.18

Table 3 The comparisons of execution time among three methods (ms)

Images	Memon et al.'s	Battiato et al.'s	Proposed scheme
Lena (64)	4526	456	1203
Peppers (64)	3659	321	1328
Logo (64)	3521	654	975
Farmer(64)	3698	578	1025
	3851	502.25	1132.75

From Table 2, we can see the average bpp of our proposed scheme is 4.18, it is close to Memon et al.'s 4.15 bpp. Sometimes, our scheme can offer lower bpp than Memon et al.'s does. Take "Pepper" for example, the bpp of our scheme is 4.18 which is lower that Memon et al.'s 4.21 bpp. Nevertheless, in the other way, our average bpp is significantly higher than Battiato et al.'s 5.04 bpp. From Table 3, we can find that the execution time of Memon et al. is also three times of our proposed scheme. Combine results of Tables 2 and 3, we can see although Memon et al. provides the best compression ratio compared with Battiato et al.'s and ours. But, it takes three times of our execution time to provide extra 0.03 compression ratio. On the contrary, our proposed scheme provides acceptable compression ratio in reasonable execution time than Memon et al.'s does.

5. Conclusions

In this paper, we presented a GA-based scheme for color re-indexing of palette-based images. In our proposed scheme, the genetic algorithm with heuristic crossover operation successfully accelerates the execution time. Experimental results show that our proposed scheme only takes one third of Memon et al.'s computation time to provide the compression ratio that is very close to what they do. When the color size is increasing, Memon et al.'s and Battoato et al.'s methods are difficult to implement because their time complexity is large. However, our algorithm can still obtain reasonable results by using acceptable computation time. Hence, our scheme is suitable for the real-time applications that require the higher compression ratio no matter their palette space is large or small.

Reference

[1].S. Battiato, G. Gallo, G. Impoco, and F. Stanco, "An Efficient Re-Indexing algorithm for Color-Mapped Images," *IEEE Transactions on Image Processing*, Vol. 13, No. 11, 2004, pp. 1419-1423.

[2].D. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison-Wesley*, Reading MA, 1989.

[3].A. C. Hadenfeldt, K. Sayood, "Compression of Color-Mapped Images," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 32, No. 3, 1994, pp. 534-541.

[4].N. D. Memon, and A. Venkateswaran, "On Ordering Color Maps for Lossless Predictive Coding," *IEEE Transaction on Image Processing*, Vol. 5, No. 11, 1996, pp. 1522-1527.

[5].M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE Transactions on Image Processing*, Vol. 9, No. 8, 2000, pp. 1309-1324.

[6].W. Zeng, J. Li, and S. Lei, "An Efficient Color Re-Indexing scheme for Palette-Based Compression," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, Sep. 2000, pp. 476–479.