harmony search algorithm with dynamic pitch adjustment rate and fret width for image compression

Saparudin
Faculty of Computer Science,
Universitas Sriwijaya
Palembang, Indonesia
saparudinmasyarif@gmail.com

Ade Kurniawan
Faculty of Computer Science,
Universitas Sriwijaya
Palembang, Indonesia
adekurniawanakbar@gmail.com

***Abstract*—the rapid growth of image capturing technology has generated the digital images, which have high resolution and large size. In general, the large sized image quality is better because it has higher color intensity, but data transmission is relatively slow or even failing. Image compression can improve the efficiency of storage space and transmission bandwidth usage. However, the image compression, which is categorized as lossy can reduce image quality. This paper implements improved harmony search (HS) algorithm for compression of color images by minimizing the reduction in image quality. The parameters of original HS algorithms including pitch adjustment rate (*PAR*) and fret width (*FW*) are fixed, whereas in improved HS algorithm, *PAR* and *FW* changed dynamically in accordance with the generation of solution vectors. Experimental results show color image compression using improved HS algorithm is better than original HS algorithm and other method.**

Keywords— Image Compression; Harmony Search Algorithm; Lossy Compression; Metaheuristics.

I. INTRODUCTION

The rapid development in the technology of digital camera has generated a digital image with high resolution and large size of file, e.g. the color image 24-bit RGB color model with size 512×512 represent 16,777,216 total number of color and use 768 kilobytes of memory. The color image representing the number of bits is typically three to four greater than grayscale image and image compression can reduce transmission time by a factor of 2 to 10 or more [1]. Image compression plays a central role in the storage and transmission of color images.

There are two types of image compression namely lossy and lossless. In lossy technique there is loss of information and therefore it is not possible to precisely restore the original image. Also the image quality will be degraded. In lossless technique no information is lost and therefore the original image can be exactly reconstructed.

An image generally has redundant data that can be removed or replaced. Image compression can reduce redundant data within an image so that its size becomes smaller in storage and lighter in data transmission process. However, the image compression that is categorized as lossy leads to reduced image quality due to redundant data which is being eliminated or replaced [2, 3]. Therefore, the quality of the compressed image becomes an important issue.

The image quality can be determined by two approaches. The first approach is to rely on human perception to compare between the image compression and the original image, but human assessment can be affected by error and also different from each other. The second approach uses peak signal to noise ratio (PSNR) [4]. PSNR is the ratio between the highest intensity image and the value of errors that affect the image representation. The larger PSNR values of an image, the better image quality would be [5].

In recent years many metaheuristic optimization techniques have been used to compress digital images, including Genetic Algorithm (GA) [6, 7, 8, 9], Particle Swarm Optimization (PSO) [10, 11], and HS algorithm [2, 3].

GA is a global search technique, which emulates natural selection and natural genetics [8, 9]. GA is able to solve many large and complex problems, resistant likely to get stuck in a local optimum, and produces a global optimal solution [8]. Nevertheless, GA also has its disadvantage that it requires setting the initial value of decision variables and considers two main factors to generate a new solution vector [6]. However, there are shortcomings in the performance of GA in which slow convergence and poor stability [12, 13].

PSO has a simple formula and the population size is smaller than GA and thus easier to implement. PSO is faster rather than GA [14]. In addition, the PSO has a constant population and the number of solutions that can be controlled. However, for complex problems PSO, it will be stuck in a local optimum that premature convergence and produce local optimum solutions [15].

HS may be viewed as a simple real-coded of the GA, because it integrates many important features of the GA like mutation, recombination, and selection [16]. HS algorithm does not require setting the initial value of decision variables and generate a new vector HS algorithm after first considering all existing vectors [2]. GA only considers two parent vectors, while HS algorithm creates a new vector, after considering all

of the existing vectors. These features increase the flexibility of the HS algorithm and yield better solutions [17]. HS contains algorithm parameters including harmony memory size (*HMS)*, harmony memory considering rate (*HMCR*), maximum improvisation (*MI*), pitch adjusting rate (*PAR*) and fret width (*FW*). *PAR* and *FW* are very important parameters in fine-tuning of optimized solution vectors and can be potentially useful in adjusting convergence rate of algorithm to optimal solution [17]. Originally *PAR* and *FW* fixed parameter values were used [18]. However, Mahdavi et al. [17] suggested that *PAR* increase linearly and *FW* decrease exponentially.

Improved HS algorithm is a variant of the HS algorithm with adjustment of the *PAR* and *FW* parameters that can improve the characteristics of fine-tuning and the convergence rate. The improved HS algorithm has been applied in several fields, mainly in the fields of engineering, application examples that spring minimal of the weight, pressure vessel design, welded beam design, and disjoint feasible region [17].

In this paper, Improved HS algorithm with dynamic *PAR* and *FW* is proposed for color image compression. The improved HS algorithm steps in general are the formulation optimization problem of image compression including initialization parameters, random tuning for harmony memory initialization, harmony improvisation, harmony memory update, and performing termination.

The rest of the paper is organized as follows. We introduce metaheuristic algorithm for image compression in Section 2. Section 3 describes HS algorithm for image compression. Section 4 describes the proposed algorithm. Some experimental simulations are performed in Section 5. Finally, a conclusion is made in Section 6.

II. Metaheuristics Algorithm for Image Compression

Metaheuristics are solution techniques that orchestrate organize an interaction between local improvement procedures and higher-level strategies to produce a process accomplished by avoidance from local optima and performing a robust search of a solution space [19]. Metaheuristic algorithms typically intend to search the best solution to an optimization problem using higher-level techniques.

The problem in lossy image compression is that it leads to the image quality being reduced due to redundant data which is being eliminated or replaced. The quantitative measurements to investigate the quality of image are using PSNR. The higher value of PSNR results in a better image quality. This problem is categorized as optimization problem thus the metaheuristic techniques can be applied. Some popular metaheuristic algorithms have been implemented in terms of image compression, such as GA, PSO and HS algorithm.

Boucetta and Melkemi [20] suggested a color image compression using Discrete Wavelet Transform (DWT) and GA. DWT method transforms the image to frequency representations which is appropriate for detecting and removing redundancies. GA optimizes high degree of correlation between the RGB planes then it is reduced by transforming them into more proper space. The results of the proposed method are found to be superior in terms of quality of the reconstructed image.

Omari and Yaichi [7] used GA for fractal image compression. They use the characteristic of rational numbers in the benefit of reducing an image size. After dividing an image to several sub-images, the best rational number representing each sub-image is selected using GA. The experiment results show competitive results with peer technique such as jpeg, png, and tiff.

Ahmadi et al. [21] proposed an image compression using PSO algorithm for optimal thresholding in the DWT of an image. A variable length-coding scheme is used to encode the results. The proposed method is tested using several standard images against other popular techniques and proved to be more efficient compared to other methods.

Vahdati et al. [6] proposed a fractal image compression algorithm based on spatial correlation in image for both range and domain pool to exploit local optima and adopt hybrid PSO with GA to explore the global optima if the local optima are not satisfied. Experiment results show that the algorithm convergent rapidly and high compression ratio.

Ismail et al. [10] proposed clustering based on adaptive lifting scheme using PSO technique for image compression. They have considered a method of changing prediction functions of the lifting scheme for each image to improve the compression ratio, gaining a more accurate prediction of pixels in the image. The proposed method exhibits better accuracy in prediction and can reduce the entropy better than the conventional prediction function on the average.

Daga and Yusiong [3] applied HS algorithm for color image compression. They mentioned that the smaller the variance of the data set implies a lower dispersion. Therefore, it is considered that manipulate the RGB components of an image with least variance would result to a compressed image with minimal loss of visual information. The original HS algorithm is used to optimize sub-images in the smallest variance value of the RGB components. Experiment result shows HS algorithm was able to compress the image with lesser visual degradation as indicated by the higher PSNR values is compared the hybrid PSO-GA technique.

III. HS Algorithm for Image Compression

A. HS Algorithm

HS algorithm is a metaheuristic optimization algorithm proposed by Geem et al. [18] and it is a technique of physical principle in the taxonomy of nature-inspired computational algorithm [22]. The algorithm is inspired by the strategy to find better harmonies, which are performed by musicians to make a beautiful music. There is a correlation between perfect harmony search and optimization problems. Musicians who are creating works of music will improvise by playing a tone randomly or based on their experience to create a perfect harmony. There are three significant components i.e. harmony memory, pitch adjusting, and randomization [23]. While the optimization problem, the optimal solution of a problem is searched by constraints and objective functions. HS algorithm has two distinguishing parameters different from other

metaheuristic algorithm: *HMCR* and *PAR* are used to generate and further modify a solution [22].

B. Improved HS Algorithm

The improved HS algorithm has proposed by Madhavi et al. [17] and it has been applied to various benchmarking and engineering optimization problems. Improved HS algorithm is using a new approach to generate vector solution as to improve the accuracy and convergence rate of original HS algorithm. In the original HS algorithm, *PAR* and *FW* are fixed each of iteration. The drawback is the number of iterations required to obtain the optimal solution becomes greater and the process becomes slower [17]. The improved HS algorithm is superior to the original HS not only in finding the optimal solution, but also in time efficiency [24].

Small *PAR* value with a large *FW* will result in the ability of the algorithm to be bad and the increase in the number of iterations needed to find the optimal solution. Although, small *FW* value at the end of the generation of fine-tuning will improve the characteristics of the solution vector, but the beginning of *FW* generation should have a greater value to force the algorithm in order to increase the diversity of the solution vector. Large *FW* with a small *FW* will lead to an increase in the best solution at the end of the generation in which the algorithm converged to an optimal solution vector [17].

The average time required by HS algorithm to resolve benchmark problems are 1956.38 CPU time, while the improved HS algorithm able to resolve more quickly with time 1616.29 CPU Time, 340 CPU time faster than HS [17].

IV. Improved HS Algorithm for Color Image Compression

In this paper, the improved HS algorithm is applied for optimizing the image quality of the compressed image. There are two main processes. The first process is preprocessing that used to determine which component from image components RGB that should be manipulated. Then, second process is compressing the original image using the improved HS algorithm.

A. Preprocessing

There are three processes in this preprocessing stage, decomposition RGB components, calculate the variance of each RGB components and determine the smallest variance of RGB components.

1. Decomposition RGB components

Decomposition of RGB color image is a process of forming three new matrixes for each component red (R), green (G), and blue (B). An input color image X sized $M \times M$ can be represented as a matrix on equation (1).

$$X = \begin{bmatrix} x_{0,0} & \cdots & x_{0,M-1} \\ \vdots & \ddots & \vdots \\ x_{M-1,0} & \cdots & x_{M-1,M-1} \end{bmatrix}, \tag{1}$$

where x are values of image pixel.

Equation (2), (3), and (4) represent the matrix of the components of red, green and blue, respectively.

$$XR = \begin{bmatrix} xr_{0,0} & \cdots & xr_{0,M-1} \\ \vdots & \ddots & \vdots \\ xr_{M-1,0} & \cdots & xr_{M-1,M-1} \end{bmatrix}, \tag{2}$$

$$XG = \begin{bmatrix} xg_{0,0} & \cdots & xg_{0,M-1} \\ \vdots & \ddots & \vdots \\ xg_{M-1,0} & \cdots & xg_{M-1,M-1} \end{bmatrix}, \tag{3}$$

$$XB = \begin{bmatrix} xb_{0,0} & \cdots & xb_{0,M-1} \\ \vdots & \ddots & \vdots \\ xb_{M-1,0} & \cdots & xb_{M-1,M-1} \end{bmatrix}, \tag{4}$$

where xr, xg, and xb are the values of matrix XR, XG, and XB, respectively.

The components of red, green and blue of image performed bitwise right shift and add to the values in X. The Equation (5), (6), and (7) are used to determine the components of red, green and blue of an image.

$$xr_{i,j} = (x_{i,j} >> 16) \,\&\, oxff\,, \tag{5}$$

$$xg_{i,j} = (x_{i,j} >> 8) \,\&\, oxff\,, \tag{6}$$

$$xb_{i,j} = x_{i,j} \,\&\, oxff\,, \tag{7}$$

where $oxff$ is hexadecimal value 255.

2. Calculate variance RGB components

After the image is decomposed, calculated variance value of each component. In a set of values, smaller variance implies less dispersion in the set of values [3]. Manipulation of RGB components that have the smallest variance value, can produce compressed image with minimal loss of visual information [3]. The variance value σr^2, σg^2, and σb^2 of red, green, and blue component respectively is determined using the following equation.

$$\sigma r^2 = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}(xr_{i,j} - \mu r)^2}{M * M}, \tag{8}$$

$$\sigma g^2 = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}(xg_{i,j} - \mu g)^2}{M * M}, \tag{9}$$

$$\sigma b^2 = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}(xb_{i,j} - \mu b)^2}{M * M}, \tag{10}$$

where μr, μg, and μb are the average values red, green, and blue of matrix component, respectively.

3. Determine the minimum variance of RGB components

After obtained the variance of each component, determine the smallest variance $Var_{\min}$ from variance value of the components.

$$Var_{\min} = \min(\mu r^2, \mu g^2, \mu b^2)\,. \tag{11}$$

Component that has the smallest variance value is taken as the selected component then will be optimized at a later stage.

The compression stage uses improved HS algorithm while the other two components are left out for further combination with new components from optimization results.

B. Optimization Using Improved HS Algorithm

There are six steps to be performed for improved HS algorithm including initialization improved HS Algorithm parameters, initialization harmony memory (*HM*), generation *PAR* and *FW*, generation a new harmony, update *HM*, and check termination criteria.

1. Initialization parameter of improved HS algorithm

Initialization parameters to be used in the optimization process are *HMS*, *HMCR*, PAR_{min}, PAR_{max}, FW_{min}, and FW_{max}. Furthermore, also set the maximum number of iteration *IN* as the termination criteria. In this study the values that will be used for each parameter based on previous studies in [3], as follows: $IN = 500$, $HMS = 5$, $HMCR = 0.95$, $PAR_{min} = 0.01$, $PAR_{max} = 0.99$, $FW_{min} = 1$, and $FW_{max} = 4$.

2. Initialization of *HM*

Vector solutions have elements that represent the 2×2 block of image pixels to the selected RGB component. The following equation to calculate the number of vector elements solution.

$$D = (M \times M)/4\,, \tag{12}$$

where D are the number of elements vector solution and $M \times M$ is the size of the image.

Each element of the vector solution will be used to represent a new component that will replace elements of 2×2 pixel block from matrix that selected i.e. *XR*, *XG*, or *XB*. Furthermore, on each of iteration, the new components will continue to be optimized based on improved HS algorithm mechanism. Following the vector solutions represent a new component.

$$XS_{i,D} = [xs_{0,0}\ xs_{0,1} \cdots xs_{i,D-2}\ xs_{i,D-1}]\,, \tag{13}$$

where *XS* is vector solution, *I* is index of vector solution, and *xs* is value of element vector solution.

Then every element of the vector solution was given the initial value with the RGB value between 0 – 255 randomly for each element.

$$xs_{i,D} = rand[0,255]\,. \tag{14}$$

Next, the vector solution generated as much as *HMS*. After all vector solutions generated would be obtained *HM*.

$$HM = \begin{bmatrix} xs_{0,0} & xs_{0,1} & \cdots & xs_{0,D-2} & xs_{0.D-1} \\ xs_{1,0} & xs_{1,1} & \cdots & xs_{1,D-2} & xs_{1,D-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ xs_{HMS-2,0} & xs_{HMS-2,1} & \cdots & xs_{HMS-2,D-2} & xs_{HMS-2,D-1} \\ xs_{HMS-1,0} & xs_{HMS-1,1} & \cdots & xs_{HMS-1,D-2} & xs_{HMS-1,D-1} \end{bmatrix} \tag{15}$$

3. Generation *PAR* and *FW*

Based on discussed in [17], to resolve the issue, a modification to the parameters *PAR* and *FW* is conducted. There are four new parameters, those are PAR_{min}, PAR_{max}, FW_{min}, FW_{max} which used to generate *PAR* and *FW*, so its value can be changed dynamically according with the number of generations, using equation (16).

$$PAR\ (gn) = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{NI} \times gn \tag{16}$$

PAR(gn) is the pitch value adjustment rate in every generation of a new harmony, PAR_{min} and PAR_{max} respectively minimum and maximum values of *PAR*, *NI* is the number of vectors generated solutions, and *gn* is a number of generations.

$$FW(gn) = FW_{\max} \exp(c, gn) \tag{17}$$

$$c = \frac{\ln(\frac{FW_{\min}}{FW_{\max}})}{NI} \tag{18}$$

FW(*gn*) is a bandwidth for each generation. FW_{max} and FW_{min} are maximum and minimum bandwidth, respectively.

4. Generation new vector solution

The next step is to generate a new vector solution which each element was given a value based on *HMCR*, *PAR*, *FW*, and ransomization (*R*). There are three rules in the generation of new vector solution:

- $(R > HMCR)$

$$R = rand[0,1]\,, \tag{19}$$

therefore new decision variables will be generated randomly with a range values of 0 – 255.

$$xkb(i) = rand[B_b(i), B_a(i)]\,, \tag{20}$$

where *xkb* is new decision variable, B_a is upper boundary, B_b is lower boundary, and *i* is index of new decision variable.

- $(R < HMCR)\ and\ (R > PAR)$

Decision variable *i – th* would be taken from *HM*.

$$L1 = \text{int}[1 + (HMS - 1)rand\,, \tag{21}$$

$$L2 = HM(L1, i), \tag{22}$$

$$xkb(i) = L2, \tag{23}$$

where $L1$ is value of selected location in *HM* randomly, $L2$ is value of decision variable which taken from *HM*, and *i* is index of *HM*.

- $(R < HMCR)\ and\ (R < PAR)$

Decision variable *i – th* which taken from *HM* will be adjusted with *FW*.

$$L3 = L2 + FW * R\varepsilon, \tag{24}$$

$$xkb(i) = L3, \tag{25}$$

where *xkb* is new decision variable, $L3$ is adjusted value of decision variable, and $R\varepsilon$ is random number ranged -1 to 1.

After the decision variable *i – th* adjusted, the next step is to check the upper and lower bounds of the decision variable. If $L3 < B_b$, therefore $L3 = B_b$, and if $L3 > B_a$,

therefore $L3 = B_a$. The lower bound and the upper bound for this compression process in a row is 0 and 255.

All of the above rules are run as many as dimensions of harmony or vector solution, from *i*=1 *to D*, where *D* is the dimension of the vector solution or number of elements in the vector solutions. The results of the new vector solution will then be substituted on a selected component. The new vector solutions matrix:

$$XKB_{i,D} = [xkb_{0,0}\ xkb_{0,1} \cdots xkb_{i,D-2}\ xkb_{i,D-1}], \quad (26)$$

where *XKB is* new vector solution, *xkb* is new decision variable, *i* is vector solution index, and *D* is number of vector solution elements.

5. Updating *HM*

Steps for updating *HM* as follows:

- Subtitute value of new RGB
  Then new solution vector is substituted to the original RGB components that previously have been selected based on the value of the smallest variance. The value of each element of harmony or the decision variables will represent the 2×2 block of pixels in the image. The following matrix is new components for image size 4×4 from the results of new solution vector:

$$XBR(i,j) = \begin{pmatrix} xkb_{0,0} & xkb_{0,0} & xkb_{0,1} & xkb_{0,1} \\ xkb_{0,0} & xkb_{0,0} & xkb_{0,1} & xkb_{0,1} \\ xkb_{0,2} & xkb_{0,2} & xkb_{i,D-1} & xkb_{i,D-1} \\ xkb_{0,2} & xkb_{0,2} & xkb_{i,D-1} & xkb_{i,D-1} \end{pmatrix}, \quad (27)$$

  where *XBR* is matrix of new component and *i, j* is new component index.

- Create new image using new component
  After the selected component is replaced by a new RGB component optimization results from the new solution vector, the new RGB components then merged back together with two previous RGB components to form a compressed image. If the selected component is red, then the equation to form each new pixel for image compression is:

$$xp(i,j) = (xbr(i,j) << 16) \mid xg(i,j) << 8 \mid xb(i,j), \quad (28)$$

  if the selected component is green,

$$xp(i,j) = (xr << 16) \mid xbr(i,j) << 8 \mid xb(i,j), \quad (29)$$

  if the selected component is blue,

$$xp(i,j) = (xr(i,j) << 16 \mid xg(i,j) << 8 \mid xbr(i,j), \quad (30)$$

  where *xp* is pixel values from merging of new components with two old component, *xbr* is pixel value of new component, *xr*, *xg*, *xb* are pixel value of red, green and blue, respectively.

- Evaluate PSNR of new image
  Quality of compression image is evaluated using PSNR. If the new vector solution which has been generated, obtained a higher PSNR compared to the worst harmony (harmony with the smallest PSNR) in *HM*.

$$PSNR = 10\log_{10}\left(\frac{255^2}{\frac{1}{M*M}\sum_{i-1}^{M}\sum_{j-1}^{M}(x(i,j)-xp(i,j))^2}\right). \quad (31)$$

  The new solution vector will be put into *HM*. while the worst harmony will be removed from the *HM*. Conversely, if the new vector solution obtained lower PSNR than the worst harmony, then the *HM* is not updated.

6. Check terminating criteria

If the terminating criteria are fulfilled, it means the stopping criterion is the number of iteration which has been initialized, then the computing process stopped in order to get the vector solution or harmony that has the best PSNR, vector solutions with the best PSNR is what will produce the compressed image. If the stopping criterion has not been fulfilled, then go back to step three.

V. Results and Discussion

The experiment using 20 RGB color images, 24 bit with bitmap file format. The image is divided into three types of image size 512×512, 256×256, and 128×128. Computer specification are Processor Intel CoreTM i5 2430M (2.4GHz, 3MB L3 cache), RAM 4 GB, Windows 7 operating system. Parameters used in experiment are compression ratio (CR), compression time (CT), and PSNR.

Table 1, Table 2, and Table 3 shows all of the experiment results of images compression with sizes that image using improved HS algorithm and original HS algorithm.

TABLE 1. Experiment Result Using Improved HS Algorithm and Original HS Algorithm for 512×512 Image Size

No	Image	Improved HS algorithm			Original HS algorithm		
		CR	CT	PSNR	CR	CT	PSNR
1	Airplane.bmp	95.13	123	32.96	94.51	182	27.3
2	Baboon.bmp	91.11	130	25.64	89.88	178	23.44
3	Barbara.bmp	94.33	126	30.63	92.11	177	26.51
4	Couple.bmp	96.86	130	39.25	96.31	160	29.14
5	Earth.bmp	94.28	134	34.65	93.85	164	27.62
6	Girl.bmp	96.66	116	39.01	96.05	159	28.78
7	Girl2.bmp	97.95	128	39.83	96.76	165	28.18
8	Girl3.bmp	96.70	133	39.25	95.95	168	28.16
9	House.bmp	96.60	136	39.33	94.76	173	28.18
10	House2.bmp	93.86	130	29.00	91.79	170	25.87
11	Island	92.66	134	33.29	92.32	173	27.33
12	Jelly.bmp	97.88	123	40.59	95.16	180	28.29
13	Jelly2.bmp	97.30	140	39.53	94.87	175	28.25
14	Lake.bmp	93.47	129	31.76	92.41	169	26.83
15	Lena.bmp	95.25	118	35.00	94.64	167	27.18
16	Peppers.bmp	94.79	129	33.45	94.29	164	27.27
17	Sandiego.bmp	89.35	119	32.05	89.01	173	25.89
18	Splash.bmp	95.96	101	33.61	95.35	165	27.46
19	Tiffany.bmp	95.23	110	34.91	94.66	171	27.33
20	Tree.bmp	95.03	112	35.21	93.75	171	27.8

Based on experiments performed on 20 color image with size 512×512 and 256×256. In *Fig.1*, and *Fig. 4*, obtained compression ratio of improved HS algorithm is greater than

original HS algorithm. *Fig. 2,* and *Fig. 5,* shows compression times of improved HS algorithm is faster than original HS algorithm. *Fig. 3,* and *Fig. 6,* shows that PSNR of improved HS algorithm greater than original HS algorithm, it means the quality of compression image using improved HS algorithm is better than original HS algorithm.

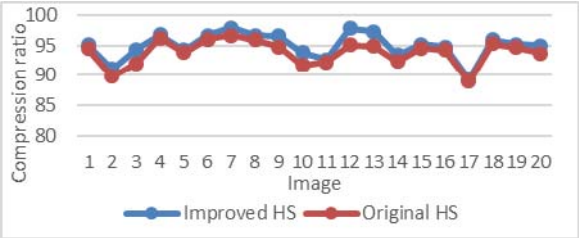

Fig. 1. CR of improved HS versus original HS for 512×512 image size

Fig. 2. CT of improved HS versus original HS for 512×512 image size

Fig. 3. PSNR of improved HS versus original HS for 512×512 image size

TABLE 2. EXPERIMENT RESULT USING IMPROVED HS ALGORITHM AND ORIGINAL HS ALGORITHM FOR 256×256 IMAGE SIZE

No	Image	Improved HS algorithm			Original HS algorithm		
		CR	CT	PSNR	CR	CT	PSNR
1	Airplane.bmp	92.93	28	30.60	92.41	46	26.73
2	Baboon.bmp	90.19	33	27.59	89.25	43	23.34
3	Barbara.bmp	92.91	29	29.15	91.18	45	25.99
4	Couple.bmp	94.90	30	33.43	94.50	44	28.32
5	Earth.bmp	91.85	36	31.00	91.40	47	26.79
6	Girl.bmp	94.77	32	33.38	94.31	45	28.04
7	Girl2.bmp	96.87	32	34.52	95.79	46	27.83
8	Girl3.bmp	94.67	29	33.76	94.24	44	27.73
9	House.bmp	94.82	31	33.86	93.51	45	27.71
10	House2.bmp	91.99	33	26.22	90.44	45	24.39
11	Island	90.05	32	30.02	89.72	46	26.43
12	Jelly.bmp	96.55	36	35.32	94.18	47	27.9
13	Jelly2.bmp	95.46	36	33.97	93.42	47	27.68
14	Lake.bmp	91.05	34	30.05	90.26	48	26.4
15	Lena.bmp	93.21	37	33.03	92.75	46	26.78
16	Peppers.bmp	92.67	31	31.53	92.31	45	26.71
17	Sandiego.bmp	87.68	34	30.12	87.31	49	25.54
18	Splash.bmp	94.36	27	30.88	93.91	46	26.78
19	Tiffany.bmp	93.98	34	33.07	93.54	47	27.53
20	Tree.bmp	91.97	33	28.57	91.07	43	25.78

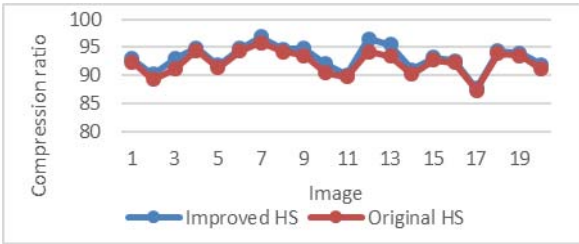

Fig. 4. CR of improved HS versus original HS for 256×256 image size

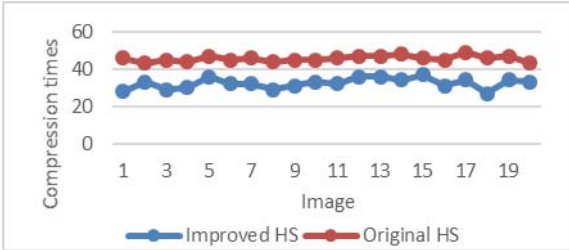

Fig. 5. CT of improved HS versus original HS for 256×256 image size

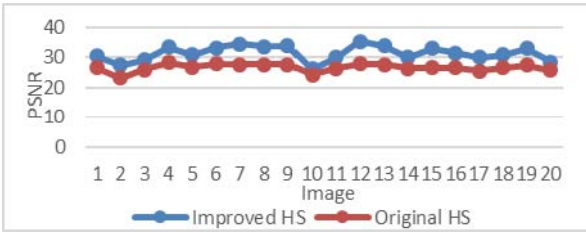

Fig. 6. PSNR of improved HS versus original HS for 256×256 image size

TABLE 3. EXPERIMENT RESULT USING IMPROVED HS ALGORITHM AND ORIGINAL HS ALGORITHM FOR 128×128 IMAGE SIZE

No	Image	Improved HS algorithm			Original HS algorithm		
		CR	CT	PSNR	CR	CT	PSNR
1	Airplane.bmp	90.21	10	28.63	89.90	14	26.23
2	Baboon.bmp	89.25	10	25.37	87.92	10	23.62
3	Barbara.bmp	89.88	8	27.28	88.50	11	25.45
4	Couple.bmp	92.73	8	31.31	91.20	10	26.94
5	Earth.bmp	89.10	9	28.87	88.44	13	25.24
6	Girl.bmp	92.08	8	30.74	91.56	14	26.37
7	Girl2.bmp	94.78	8	32.88	93.59	15	26.51
8	Girl3.bmp	92.70	8	30.12	91.16	15	25.68
9	House.bmp	92.41	8	30.24	90.94	14	25.67
10	House2.bmp	89.98	9	25.00	88.44	11	23.22
11	Island	87.52	8	28.41	86.15	12	25.04
12	Jelly.bmp	94.06	8	30.46	91.66	15	25.75
13	Jelly2.bmp	91.94	8	28.85	90.16	12	25.17
14	Lake.bmp	87.76	8	27.94	84.14	14	24.81
15	Lena.bmp	90.04	8	29.49	89.56	15	25.31
16	Peppers.bmp	88.90	7	29.19	88.44	12	24.74
17	Sandiego.bmp	86.62	8	29.27	85.20	14	25.61
18	Splash.bmp	91.89	7	28.06	90.32	13	24.82
19	Tiffany.bmp	91.54	8	31.86	90.09	11	26.45
20	Tree.bmp	88.14	8	25.13	86.29	12	23.18

Furthermore, improved HS algorithm is compared to GA for image compression. Based on experiments are performed on the image size of 512×512 using GA algorithm in [25]. While the experiment performed on the image size of 512×512 using improved HS algorithm, PSNR much higher

than the GA algorithm, which is the quality of image compression using improved HS algorithm is much better than using the GA algorithm. Table 4 shows the improved HS algorithm and GA.

TABLE 4. COMPARISON PSNR OF IMPROVED HS ALGORITHM AND GA FOR IMAGE COMPRESSION

No.	Image	GA	Improved HS
1	Airplane.bmp	26.23	32.96
2	House.bmp	23.62	39.33
3	Lenna.bmp	25.45	35.00
4	Splash.bmp	26.94	33.61
5	Jelly.bmp	25.24	40.59
6	Jelly2.bmp	26.37	39.53
7	Peppers.bmp	26.51	33.45
8	Tree.bmp	25.68	35.21

VI. SUMMARY AND CONCLUSIONS

Image compression using improved HS algorithm has been performed on three types of image size. The experiment shows that three parameters; CR, CT, and PSNR image compression using improved HS algorithm becomes more powerful compared to original HS algorithm. At last, improved HS algorithm is compared to GA in performing image compression, then produces the compressed image quality which is better than GA. The improvement in the process of block of image optimization will be needed, in order to accelerate time compression by modifying parameter values of *HMCR* and *HM*.

REFERENCES

[1] Gonzalez, R.C. and Woods, R.E. (2008). Digital Image Processing, Third Edition. Pearson Prentice Hall.

[2] Bansod, S., & Jain, S. (2014). Improved harmony Search Algorithm for Color Image Compression. International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 2, 2014 (hal. 1669-1672). IJRITCC.

[3] Daga, R.R.M., & Yusiong, J.P.T. (2012). Image Compression Using harmony Search Algorithm. International Journal of Computer Science Issues, Vol. 9, 2012 (hal. 16-23). IJCSI.

[4] Marimuthu, M., Muthaiah. R., & Swaminathan. P. (2012). Review Article : An Overview of Image Compression Techniques. Research Journal of Applied Sciences, Engineering And Technology, Vol 4, 2012 (pp. 5381-5386). Maxwell Scientific Organization.

[5] Kaushik, P., & Sharma. Y. (2012). Comparison Of Different Image Enhancement Techniques Based Upon PSNR & MSE. International Journal of Applied Engineering Research, Vol 7, No 11, 2012. Research India Publications.

[6] Vahdati, G., Khodadadi, H., Yaghoobi, M., & Akbarzadeh, R.M. (2010). Fractal Image Compression Based on Spatial Correlation and Hybrid Particle Swarm Optimization with Genetic Algorithm. 2nd International Conference on Software Technology and Engineering (ICSTE), Vol. 2, 2010 (pp. 185-189). IEEE.

[7] Omari, M., and Yaichi, S. (2015). Image Compression Based on Genetic Algorithm Optimization. 2nd World Symposium on Web Applications and Networking (WSWAN), pp. 1-5. IEEE.

[8] Wu, M.S., & Lin, Y.L. (2010). Genetic Algorithm with A Hybrid Select Mechanism for Fractal Image Compression. Digital Signal Processing, Vol. 20, 2010 (pp 1150-1161). Elsevier Science Ltd.

[9] Xi, L., and Zhang, L. (2007). A Study of Fractal Image Compression Based on an Improved Genetic Algorithm. International Journal of Nonlinear Science, Vol.3, No.2, pp. 116-124.

[10] Ismail, M.M., & Baskaran. K. (2010). Clustering Based Adaptive Image Compression Scheme Using Particle Swarm Optimization Technique. International Journal of Science and Technology, Vol 2, 2010 (pp. 5114-5119).

[11] Kanvel, N., Letitia, S., Monie, E.C. (2010). Adaptive Lifting Based Image Compression Scheme with Particle Swarm Optimization Technique. International Journal of Engineering Science and Technology, Vol. 2, No. 9, pp. 4886-4895.

[12] Fogel, D.B. (2005). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 3rd Edition. Wiley-IEEE Press.

[13] Gao, W. (2003). An Improved Fast-Convergent Genetic Algorithm. Proceeding on IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, Vol 2, pp. 1197-120.

[14] Urade, H.S. and Patel, R. (2011). Study and Analysis of Particle Swarm Optimization: A Review. IJCA Proceedings on 2nd National Conference on Information and Communication Technology NCICT, pp. 1-5.

[15] Shindu, M., Rajkamal, R. (2009). Image and Its Compression Techniques – A review. International Journal of Recent Trends in Engineering, Vol. 2, No. 2, pp.71-75. Academy Publisher.

[16] Chakraborty, P., Roy, G.G., Das, S., Jain, D., and Abraham, A.(2009). An Improved Harmony Search Algorithm with Differential Mutation Operator. Fundamenta Informaticae, Vol. 95, pp. 1–26.

[17] Mahdavi, M., Fesanghary. M., & Damangir. E. (2007). An Improved Harmony Search Algorithm for Solving Optimization Problems. Applied mathematics and Computation, Vol 188, 2007 (pp. 1567-1579). Elsevier Science Ltd.

[18] Geem, Z.W., Kim, J.H., & Loganathan, G.V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. Transactions of The Society fo Modeling and Simulation International (Simulation), Vol. 76, Issue. 2, 2001 (pp. 60-68). United States of America : Simulation Councils Inc.

[19] Gover, F. and Kochenberger, G.A. (2003). Handbook of Metaheuristics. Kluwer Academic Publishers.

[20] Boucetta, A. and Melkemi, K.E. (2012). DWT Based-Approach for Color Image Compression Using Genetic Algorithm. Book Chapter: Image and Signal Processing, Volume 7340 of the series Lecture Notes in Computer Science pp 476-484.

[21] Ahmadi, K., Javaid, A.Y., and Salari, E. (2015). An Efficient Compression Scheme based on Adaptive Thresholding in Wavelet Domain Using Particle Swarm Optimization. Journal Image Communication, Elsevier Science, Vol. 32, Issue C, pp 33-39.

[22] Wang, X., Zhi Gao, X., Zenger, K. (2015). An Introduction to Harmony Search Optimization Method. Springer Brief in Applied Sciences and Technology – Computational Intelegence. Springer.

[23] Geem ZW, Kim JH and Loganathan GV. (2001). A New Heuristic Optimization Algorithm: Harmony Search. Simulation, Vol. 76, No. 2, pp 60-68.

[24] Madivada, H., & Rao. C.S.P. (2013). Improved Music Based Harmony Search For Solving Job Shop Scheduling. International Journal of Programming Languages And Applications, Vol 3, No 3, 2013. IJPLA.

[25] Idrees, A, K., Ali, S, A., & Obead, E,H. (2012). Image Compression Using Genetic Algorithm. Journal of Babylon University, Vol 20, No 2, 2012.