A novel Equilibrium Optimization Algorithm for Multi-thresholding Image Segmentation Problems

Mohamed Abdel-Basset[1], Victor Chang[2] and Reda Mohamed[1]
[1] Faculty of Computers and Informatics, Zagazig University, Sharqiyah, Egypt.
E-mails: analyst_mohamed@zu.edu.eg; redamoh@zu.edu.eg
2. School of Computing, Engineering and Digital Technologies, Teesside University, UK
Email: victorchang.research@gmail.com/V.Chang@tees.ac.uk
*Correspondence: Mohamed Abdel-Basset; Email: analyst_mohamed@zu.edu.eg; Tel: +20112228843

Abstract

Image segmentation is considered a crucial step required by image analysis and research. Many techniques have been proposed to resolve existing problems and improve the quality of research, such as Region-based, threshold-based, edge-based, and feature-based clustering in the literature. The researchers have moved towards using the threshold technique due to the ease of use for image segmentation. To find the optimal threshold value for a grayscale image, we improved and used a novel meta-heuristic equilibrium algorithm to resolve this scientific problem. Additionally, our improved algorithm has the ability to enhance the accuracy of the segmented image for research analysis with a significant threshold level. The performance of our algorithm is compared with seven other algorithms like whale optimization algorithm (WOA), bat Algorithm (BA), sine-cosine algorithm (SCA), salp swarm algorithm (SSA), harris hawks algorithm (HHA), crow search algorithm (CSA), and particle swarm optimization (PSO). Based on a set of well-known test images taken from Berkeley Segmentation Dataset (BSD), the performance evaluation of our algorithm and well-known algorithms described above has been conducted and compared. According to the independent results and analysis of each algorithm, our algorithm can outperform all other algorithms in fitness values, Peak signal to noise ratio (PSNR) metric, Structured similarity index metric (SSIM), maximum absolute error (MAE), and and Signal to noise ratio (SNR). However, our algorithm cannot outperform on some algorithms in standard deviation (Std) values, and central processing unit (CPU) time with the large thresholds level observed.

Keywords: Image segmentation problem**,** equilibrium optimization algorithm (EOA), and Kapur's entropy

1. Introduction

Image segmentation is considered an important step required by image research, so that research analysis can be performed accurately. The main reason is that image segmentation can subdivide an image into several non-overlapping homogenous regions, in order to facilitate a better image analysis [1]. Image segmentation is instrumental for many computer vision applications, such as medical imaging, robotic vision, biomedical image processing, pattern recognition, and so on. Several algorithms have been proposed for image segmentation based on one of the following techniques:

- Region-based
- Edge-based

- Threshold-based
- Feature-based clustering

Based on comparison with literature, thresholding-based segmentation is considered as the preferred technique due to its ease to use, small storage space required, accuracy and speed [1-3]. Thresholding is classified into two classes: bi-level and multi-level. In bi-level thresholding, one threshold value is used to divide the image into two homogenous areas, foreground, and background. But when the image contains more than two homogenous regions, bi-level thresholding is unable to separate those regions. Hence, there is a strong necessity for multi-level thresholding to subdivide an image into the number of areas of homogenous pixels based on maximizing the methods illustrated later. Fundamentally, the optimal threshold values can be obtained using one of the following two approaches: parametric and nonparametric [4].

In a parametric approach, some statistical parameters should be calculated for each class in the image. In relations to the parametric approach, the optimal values can be found by using the combined methods, such as using Otsu's method [5] and Kaptur's entropy [6]. Otsu's method work on separating the regions that contain the pixel intensities close to each other based on maximizing the between-regions variance. On the other hand, Kapur's method attempts to find the homogenous regions based on maximizing the variance in an image. The maximum variance is also called maximum entropy to maximize the entropy of the areas.

All the methods discussed above, proposed for finding the optimal thresholds effectively, include bi-level thresholding. However, it is unable to achieve the optimal threshold values for multi-level thresholding, as well as the time complexity of those methods grow exponentially when the number of threshold level increases significantly. Therefore, new and effective methods are required by the multi-level thresholding problem, especially for those who have a massive number of image thresholds. With the improvement of meta-heuristic algorithms and their successful examples in solving ongoing optimization problems [7-9], the researchers confirmed meta-heuristic algorithms can be used to resolve the multi-thresholding problem.

In the last decades, many meta-heuristic algorithms, such as genetic algorithm [10], particle swarm optimization(PSO) [11], Whale optimization algorithm(WOA) [12], ant-colony optimization algorithm (ACO) [13], firefly optimization (FFA) [14], honey bee mating optimization (HBM) [15], cuckoo search (CS) [16], flower pollination Algorithm (FPA) [17], symbiotic organisms search (SOS) [18], and Moth-flame optimization (MFA) [12] have been proposed for tackling the multi-thresholding problems based on maximizing some specific criteria for research.

In [11, 12], the authors proposed maximizing Otsu's criterion for tackling image segmentation by using PSO, and WOA. The authors do not check the performance of the WOA on large threshold levels to check the stability of its performance on high-dimensional levels. But based on our experiments in the experiment section, it shows that The PSO and WOA still suffer from local optima with the large thresholds level. Also, in [12], Moth-flame optimization is applied in segmenting the images based on maximizing Otsu's method. Additionally, the authors do not check the performance of the Moth-flame optimization on the large thresholds level. Whereas, the author in [14] developed FFA for tackling a multi-level thresholding image segmentation problem, and the improved one (IFFA) in [19] attempts to improve the performance of FFA to resolve the multi-thresholding problem. The Cauchy mutation and neighborhood strategy is used in IFFA to increase the exploration operation and accelerate the convergence. Also, the authors didn't evaluate the performance of both FFA, IFFA on the large thresholds level.

Additionally, the author [16] proposed the cuckoo search algorithm for tackling the multi-thresholding problem based on maximizing the tsallis entropy. The authors test the performance of the cuckoo search algorithms on small thresholds level reaches 5 and don't test its performance on high thresholds level. In [18], the author proposed an improved SOS (ISOS) to tackle the multi-thresholding image segmentation problem for the color images. ISOS used The opposite based learning strategy to accelerate the convergence and enhance the performance of the original SOS. Moreover, the modified ABC (MABC) in [20], was proposed for segmenting the satellite image using various fitness functions. The authors found that ABC has a poor performance in the exploitation phase, so they proposed the enhanced version (MABC) by introducing a new initialization strategy using a chaotic search and a novel search technique using differential evolution.

In [21], the improved PSO is adapted for segmenting the cancer infected breast thermal images by maximizing Otsu's method. In addition, the author in [22], proposed the real coded genetic algorithm with simulated binary crossover (SBX) for segmenting the medical brain images by maximizing the Kapur's entropy. In [23], the authors proposed method consist of two phases, in the first phase Genetic algorithms (GA) is used to determine the execution sequence of the meta-heuristic algorithms, and the second phase contains four meta-heuristic algorithms which are executed in a specific order based on the current solution of the GA.

In this paper, our improvement of another novel meta-heuristic algorithm, namely Equilibrium optimizer (EO) [24], is developed for tackling the multi-thresholding problems. EO is based on dynamic mass balance on a control volume, where it uses a mass balance equation to measure the number of mass entries and be generated in the volume over a period of time and seek to find the state that achieves the equilibrium of the system. EO has several advantages qualified as a good algorithm, such as the ability on proposing a balance between the exploration and exploitation operators, easy to implement, and the diversity among the individuals in a population. As a result, it can be used for solving many optimization problems in the real world, such as DNA fragment assembly problem, flow shops scheduling problems, and so on. Here, we will examine the performance of our improved EO algorithm on multi-thresholding image segmentation problems.

The main contributions of our paper include:

- Adapting EO for solving the image segmentation problem.
- Testing the performance of our proposed algorithms on large threshold levels reaches 50 threshold values to check the stability of its performance.

- Comparing the performance of our proposed algorithm with seven other algorithms to evaluate its performance compared with those algorithms.
- Our algorithm can outperform all other algorithms in fitness values, PSNR, SSIM, MAE, and SNR.

The rest of this paper is organized as follows. Section 2 describes the mathematical model of Kapur's method. Section 3 summarizes the equilibrium optimizer. Section 4 presented the proposed approach. Section 5 presented the discussion and the experimental results of the proposed approach for tackling multi-level thresholding on 7 test images. Section 6 demonstrates some conclusions about the proposed approach and future work.

2. Kapur's entropy method

This method seeks for finding the optimal threshold value, t. Generally, t takes a value between 1 and 255 (for 8-bit depth images), that subdivides an image into E_0, and E_1 until maximizing the following function:

$$F(t) = E_0 + E_1 \tag{1}$$

$$E_0 = -\sum_{i=0}^{t-1} \frac{X_i}{T_0} * \ln \frac{X_i}{T_0},\ X_i = \frac{N_i}{T}, T_0 = \sum_{i=0}^{t-1} X_i \tag{2}$$

$$E_1 = -\sum_{i=t}^{L-1} \frac{X_i}{T_1} * \ln \frac{X_i}{T_1},\ X_i = \frac{N_i}{T}, T_1 = \sum_{i=1}^{t-1} X_i \tag{3}$$

Where N_i is the number of pixels with the grey value, i, and T is the number of pixels in an image. Eq.1 can be adapted easily for finding multi-threshold values that separate the image into several homogenous regions, where it is redesigned as follows:

Assume that a grey image with intensity values in the range [0, L-1] is given, then this method seeks to find n optimal threshold values $[t_0, t_1, t_2, \dots \dots .., t_n]$ that subdivide the image to $[E_0, E_1, E_2, \dots \dots .., E_n]$ to maximize the following function:

$$F(t_0, t_1, t_2, \dots \dots .., t_n) = E_0 + E_1 + E_2 + \cdots \dots .. + E_n \tag{4}$$

$$E_0 = -\sum_{i=0}^{t_0-1} \frac{X_i}{T_0} * \ln \frac{X_i}{T_0},\ X_i = \frac{N_i}{T}, T_0 = \sum_{i=0}^{t_1-1} X_i \tag{5}$$

$$E_1 = -\sum_{i=t_0}^{t_1-1} \frac{X_i}{T_1} * \ln \frac{X_i}{T_1},\ X_i = \frac{N_i}{T}, T_1 = \sum_{i=t_0}^{t_1-1} X_i \tag{6}$$

$$E_2 = -\sum_{i=t_1}^{t_2-1} \frac{X_i}{T_2} * \ln \frac{X_i}{T_2},\ X_i = \frac{N_i}{T}, T_2 = \sum_{i=t_1}^{t_2-1} X_i \tag{7}$$

$$E_n = -\sum_{i=t_n}^{L-1} \frac{X_i}{T_n} * \ln \frac{X_i}{T_n},\ X_i = \frac{N_i}{T}, T_n = \sum_{i=t_n}^{L-1} X_i \tag{8}$$

3. Equilibrium optimizer

In [24], another meta-heuristic algorithm inspired by the physics laws, namely equilibrium Optimizer (EO), is proposed for solving the optimization problems, with some descriptions discussed at the end of Section 1. More information on the inspiration of EO is found in [24]. The mathematical model of EO algorithm is illustrated in the following three steps:

Step 1: initialization

In this step, EO uses a group of particles, in where each particle represents the concentration vector that contains the solution for the optimization problem. The initial concentrations vector is generated randomly in the search space using the following formula:

$$\vec{v}_{i=} c_{min} + (c_{max} - \mathrm{c}_{min}) * \mathrm{r} \quad \mathrm{i} = 0,1,2, \ldots, \mathrm{n} \tag{9}$$

Where, $\vec{\mathrm{v}}_{\mathrm{i}}$ represents the concentration vector of the particle i, $\mathrm{c}_{\mathrm{min}}, \mathrm{c}_{\mathrm{max}}$ determine the upper, and lower bound for each dimension in the problem, respectively, r is a random number in the range of [0,1], and n specifies the number of particles in the group.

Step 2: Equilibrium pool and candidates ($\vec{\mathrm{p}}_{\mathrm{eq,pool}}$)

For all meta-heuristic algorithms, there is an objective for each one tries to achieve it based on its nature. For example, in [25], WOA searches for prey. In [26], artificial bee colony (ABC) searches for a food source, and relative to EO, it searches for the equilibrium state of the system. When getting to the equilibrium state, EO may be getting to the near-optimal solution of the optimization problem. In the optimization process, EO does not know the level of concentrations that achieve the equilibrium state. Hence, it assigns the best four particles found in the population at equilibrium candidates plus another one containing the average of the best four particles. These five equilibrium candidates help EO in the exploration and exploitation operator, where the first four candidates help EO to have better diversification capability, and the average improvement in the exploitation. These five candidates are stored in a vector, namely Equilibrium pool:

$$\vec{\mathrm{p}}_{\mathrm{eq,pool}} = \left[\vec{\mathrm{p}}_{\mathrm{eq}\,(1)},\ \vec{\mathrm{p}}_{\mathrm{eq}\,(2)},\ \vec{\mathrm{p}}_{\mathrm{eq}(3)},\ \vec{\mathrm{p}}_{\mathrm{eq}(4)},\ \vec{\mathrm{p}}_{\mathrm{eq(avg)}}\right] \tag{10}$$

More information on the equilibrium candidates is found in [24].

Step 3: updating the concentration

The following term helps EO having a plausible balance between intensification and diversification. Since turnover rate can vary over time in a real control volume, $\vec{\lambda}$ is supposed to be a random vector between 0 and 1.

$$\vec{F} = e^{-\vec{\lambda}(t-t_0)} \tag{11}$$

Where, t is decreased with the increment of the iteration (it) using the following formula:

$$t = \left(1 - \frac{it}{t_{max}}\right)^{\left(a2*\left(\frac{it}{t_{max}}\right)\right)} \tag{12}$$

Where it, and t_{max} is the current and the maximum iterations, respectively. And $a2$ is a constant value used to control the intensification (Exploitation) capability. Another factor, $a1$, is used to improve the diversification and intensification of EO and is formed as follows:

$$\vec{t}_0 = \frac{1}{\vec{\lambda}} \ln \left(-a_1 sign(\vec{r} - 0.5)[1 - e^{-\vec{\lambda}t}\right) + t \tag{13}$$

Where a_1 is a constant value used to manage the exploration capability when a_1 is higher, the diversification capability is better and the intensification capability is lower. In contract to a_1, a_2 is a constant value used to control the exploitation capability. When a_2 is higher, the intensification capability is better and the diversification capability is lower. Generation rate (R) is another term used to improve the intensification operator and is formulated as follows:

$$\vec{R} = \vec{R}_0 * e^{-\vec{\lambda}*(t-t_0)} \tag{14}$$

Where $\vec{\lambda}$ is a random vector in the range of [0, 1], and $\vec{R}_0$ is the initial value and is formulated as follows:

$$\vec{R}_0 = \overrightarrow{RCP} * \left(\overrightarrow{c_{eq}} - \vec{\lambda} * \vec{C}\right) \tag{15}$$

$$\overrightarrow{RCP} = \begin{cases} 0.5r_1 & r_2 > RP \\ 0 & otherwise \end{cases} \tag{16}$$

Where r_1 and r_2 are random numbers between 0 and 1. In this equation, $\overrightarrow{RCP}$vector is the generation rate control parameter that determines whether the generation rate will apply to the updating process based on a probability RP. Finally, the updating equation of EO is as follows:

$$\vec{C} = \overrightarrow{c_{eq}} + \left(\vec{C} - \overrightarrow{c_{eq}}\right) * \vec{F} + \frac{\vec{R}}{\vec{\lambda}*V} * (1 - \vec{F}) \tag{17}$$

Where V is equal to 1. For more information about EO, go to [24]. Algorithm 1 displays the steps of the equilibrium optimization algorithms to solve the optimization problems.

Algorithm 1 The Equilibrium optimizer (EO)
1. Initialize the population of particles $p_i (i = 1,2,3, \ldots, n)$
2. Set the fitness value of the four particles in equilibrium pool, p_{eq}, with a large value
3. Set parameter's value $a_1 = 1; a_2 = 2; GP = 0.5;$
4. while (it < $t_{maxIter}$)
5. for each i particle
6. Calculate the fitness value of particle i $f(\vec{p}_i)$
7. **if** $(f(\vec{p}_i) < f(\vec{p}_{eq\,(1)}))$)
8. Set $\vec{p}_{eq\,(1)}$ with $\vec{p}_i$ and $f(\vec{p}_{eq\,1})$ with $f(\vec{p}_i)$
9. **elseIf** $(f(\vec{p}_i) > f(\vec{p}_{eq\,(1)})$ and $f(\vec{p}_i) < f(\vec{p}_{eq\,(2)}))$
10. Set $\vec{p}_{eq\,(2)}$ with $\vec{p}_i$ and $f(\vec{p}_{eq\,(2)})$ with $f(\vec{p}_i)$

```
        elseif (f(p_i) > f(p_eq(2)) and f(p_i) < f(p_eq3))
            Set p_eq(3) with p_i and f(p_eq(3)) with f(p_i)
        elseif (f(p_i) > f(p_eq(3)) and f(p_i) < f(p_eq(4))))
            Set p_eq(4) with p_i and f(p_eq(4)) with f(p_i)
        end if
    end for
    p_eq(avg) = ( p_eq(1) + p_eq(2) + p_eq(3) + p_eq(4))/4
   The equilibrium pool p_eq,pool = [ p_eq(1), p_eq(2), p_eq(3), p_eq(4), p_eq(avg)]
    Accomplish the memory saving
    Assign t using Eq. (12)
    for each i particle
        Choose one candidate from p_eq,pool
        Generate two vectors, namely r, λ randomly
        Construct F using Eq. (11)
        Construct RCP using Eq. (16)
        Construct R_0 using Eq. (15)
        Construct R using Eq. (14)
        Update the concentrations using Eq. (17)
    end for
    it++
end while
```

4. The proposed algorithm

In this section, the equilibrium optimization algorithm is developed for solving multi-thresholding image segmentation problems. The steps of the proposed algorithm have been illustrated in the next subsections.

4.1. Initialization

In this phase, a population compound of a number of particles N is proposed, where each particle consists of n dimensions that are initialized randomly within the boundaries of gray levels of the image as follows:

$$S_{i,j} = H_{min} + rand(0,1) * (H_{max} - H_{min}) \tag{18}$$

Where $H_{min}, and\ H_{max}$ is the minimum and maximum of the gray level values in the image histogram, and $rand(0,1)$ is a random number in the range [0, 1].

4.2 Fitness function

The fitness function is considered an essential step in all meta-heuristic algorithms. For solving Multi-thresholding problems using EO, it must be provided with a function to evaluate the solution given by each particle in the group. Based on the nature of problems, the optimal threshold values could be obtained by optimizing the otsu's method, Tsallis entropy, and Kapure's method. In this paper, we used Kapure's method as a fitness function for evaluating the solutions. The Mathematical model of Kapure's method illustrated before.

4.3 Proposed algorithm

The steps of adapting the standard equilibrium optimization to overcome multi-thresholding problems are illustrated in Algorithm 2. Also, the flowchart of the same steps is introduced in Fig 1.

Algorithm 2 The proposed algorithm

Initialize the population of particles $\mathrm{p}_\mathrm{i} (i = 1,2,3, \ldots, n)$
Set fitness value of the four particles in equilibrium pool , p_{eq}, a large value
Set parameter's value $a_1 = 1; a_2 = 2; GP = 0.5;$
while (it < $\mathrm{t}_{\mathrm{maxIter}}$)
  for each i particle
    If $(f(\vec{\mathrm{p}}_\mathrm{i}) < f(\vec{\mathrm{p}}_{\mathrm{eq}\,(1)})))$
      Set $\vec{\mathrm{p}}_{\mathrm{eq}\,(1)}$ with $\vec{\mathrm{p}}_\mathrm{i}$ and $\mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(1)})$ with $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i})$
    elseIf $(\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) > \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(1)})$ and $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) < \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(2)}))$
      Set $\vec{\mathrm{p}}_{\mathrm{eq}\,(2)}$ with $\vec{\mathrm{p}}_\mathrm{i}$ and $\mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(2)})$ with $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i})$
    elseif $(\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) > \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(2)})$ and $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) < \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(3)}))$
      Set $\vec{\mathrm{p}}_{\mathrm{eq}\,(3)}$ with $\vec{\mathrm{p}}_\mathrm{i}$ and $\mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(3)})$ with $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i})$
    elseif $(\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) > \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(3)})$ and $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i}) < \mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(4)})))$
      Set $\vec{\mathrm{p}}_{\mathrm{eq}\,(4)}$ with $\vec{\mathrm{p}}_\mathrm{i}$ and $\mathrm{f}(\vec{\mathrm{p}}_{\mathrm{eq}\,(4)})$ with $\mathrm{f}(\vec{\mathrm{p}}_\mathrm{i})$
    End if
  End for
  $\vec{\mathrm{p}}_{\mathrm{eq(avg)}} = (\vec{\mathrm{p}}_{\mathrm{eq}\,(1)} + \vec{\mathrm{p}}_{\mathrm{eq}\,(2)} + \vec{\mathrm{p}}_{\mathrm{eq}\,(3)} + \vec{\mathrm{p}}_{\mathrm{eq}\,(4)})/4$
  The equilibrium pool $\vec{\mathrm{p}}_{\mathrm{eq,pool}} = [\vec{\mathrm{p}}_{\mathrm{eq}\,(1)}, \vec{\mathrm{p}}_{\mathrm{eq}\,(2)}, \vec{\mathrm{p}}_{\mathrm{eq}(3)}, \vec{\mathrm{p}}_{\mathrm{eq}(4)}, \vec{\mathrm{p}}_{\mathrm{eq(avg)}}]$
  Accomplish the memory saving
  Assign t using Eq. (12)
  for each i particle
    Choose one candidate from $\vec{\mathrm{p}}_{\mathrm{eq,pool}}$
    Generate two vectors, namely $\vec{\mathrm{r}}, \vec{\lambda}$ randomly
    Construct $\vec{\mathrm{F}}$ $using$ Eq. (11)
    Construct $\overrightarrow{\mathrm{RCP}}$ $using$ Eq. (16)
    Construct $\overrightarrow{\mathrm{R}_0}$ $using$ Eq. (15)
    Construct $\vec{\mathrm{R}}$ $using$ Eq. (14)
    Update the concentrations using Eq. (17)
  end for
  Calculate the fitness values for each particle using Eq. (4)
  update the equilibrium pool if there is particle better
  it++
end while

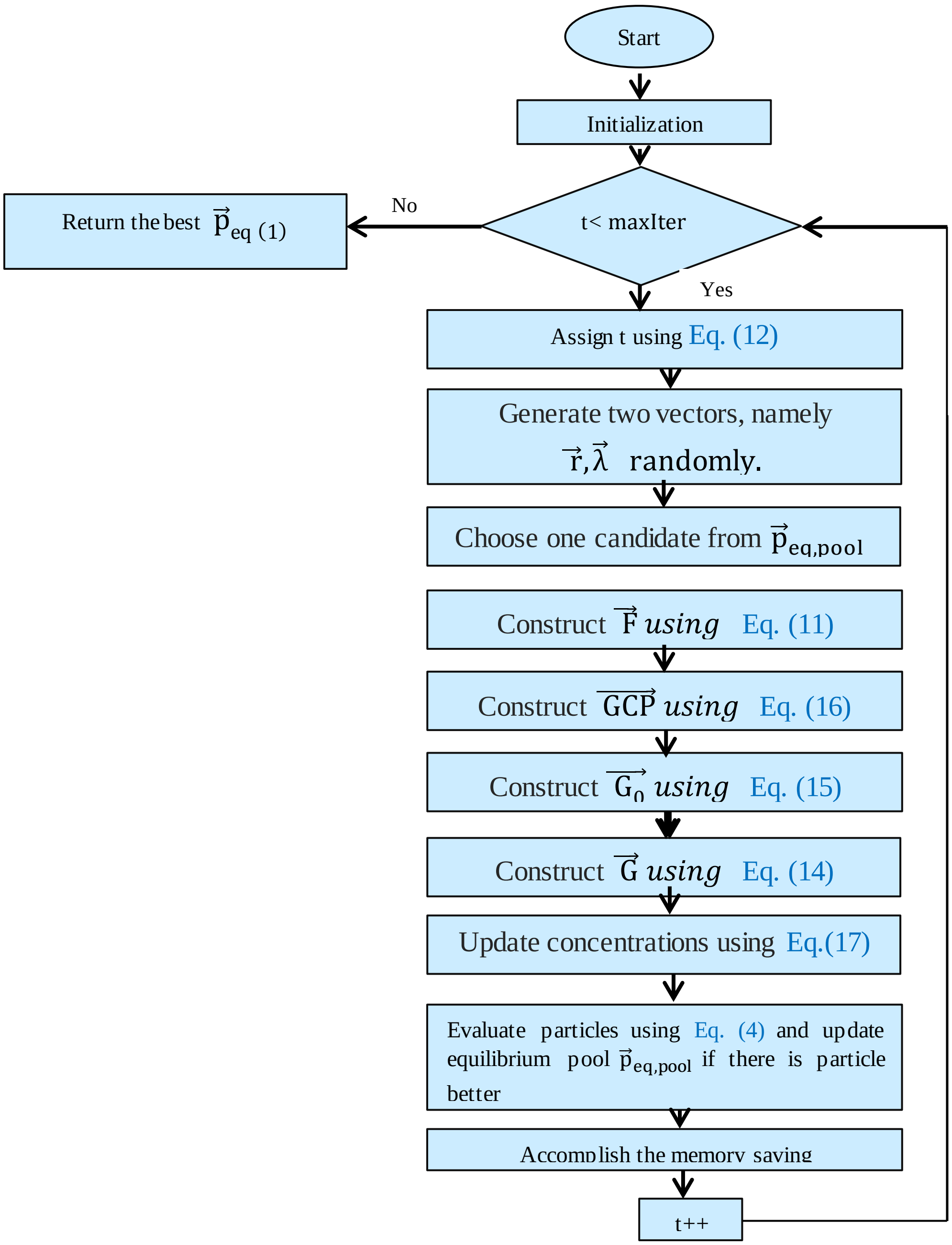


Fig 1. The steps of the equilibrium optimization for solving the multi-thresholding problem

5 Experiments and discussion

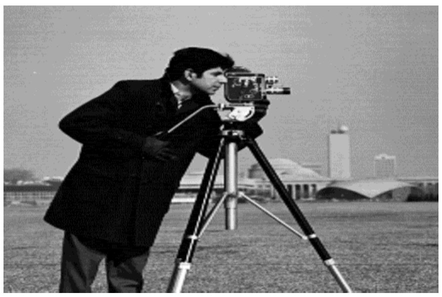
In this section, the experiment settings used by the proposed algorithm and other algorithms are presented. Moreover, the analysis of the results obtained by the proposed algorithm in terms of complexity time, quality of the results, and the statistical analysis used for comparing the proposed algorithm with other algorithms are also demonstrated here. This section is organized as follows:

1. Section 5.1: describes benchmark dataset images using in our experiments.
2. Section 5.2: illustrates experiment settings.
3. Section 5.3: shows the results of our algorithm on the benchmark dataset images.

4. Section 5.4: presents a comparison of our proposed algorithm with seven other algorithms proposed for solving this problem on 7 test images.

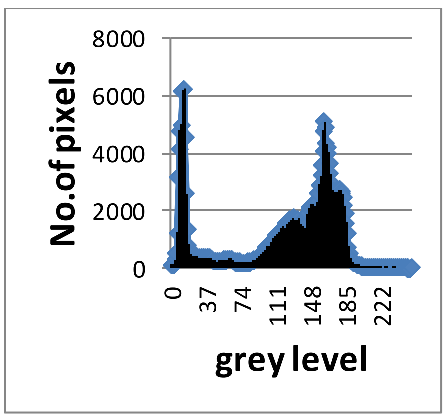
5.1 Benchmark datasets.

Our experiments used seven test images from Berkley University dataset for testing the performance of our algorithm, namely cameraman, house, lena, lake, jetplane, livingroom, and peppers. All those images are of size 512*512. The original images and the histogram of each one are shown in Fig.2.

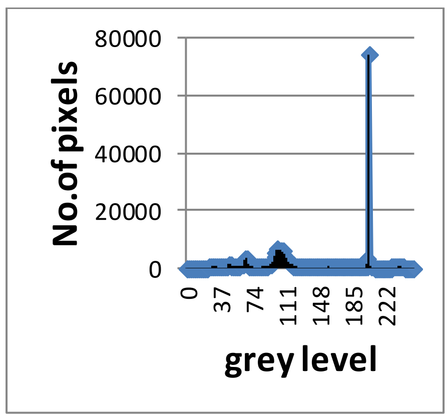

a. Original cameraman image


c. Original house image

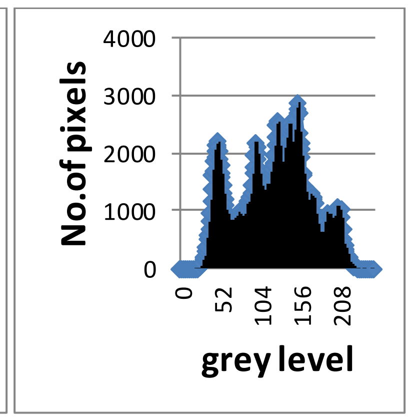

e. Original lena image



b. Cameraman image histogram



d. house image histogram

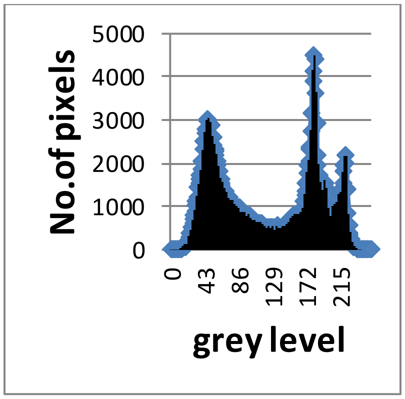


f. Lena image histogram

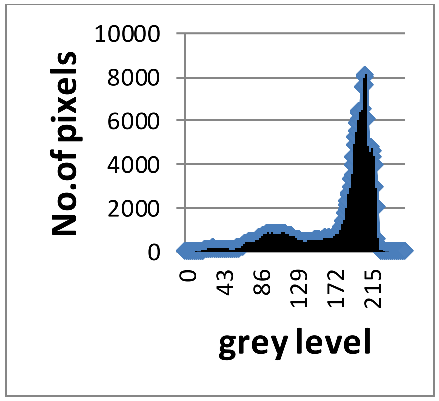

g. Original lake image


i. Original jetplane image

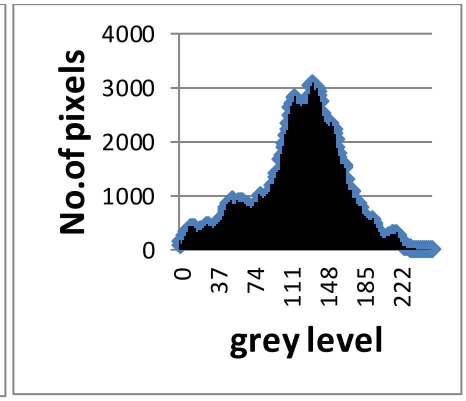

k. Original livingroom image



h. Lake image histogram

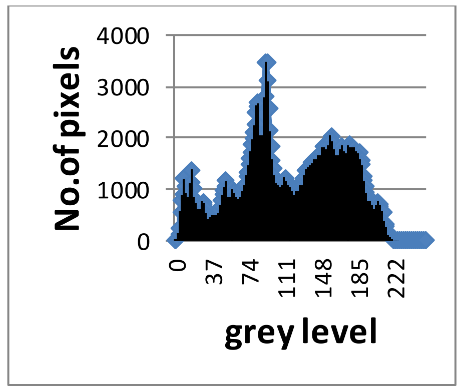


j. Jetplane image histogram



l. Livingroom image histogram

m. Original peppers image



n. Peppers image histogram

Fig.2 The original images and histogram of each test image

5.2 Experimental settings

We perform all the experimental studies on a desktop computer equipped with Windows 7 ultimate platform with a 32-bit operating system, Intel® Core™ i3-2330M CPU @ 2.20 GHz, and 1 GB of RAM. We use a low memory capacity to test our proposal can work under the most constraint conditions. All algorithms are implemented using the Java programming language. for evaluating the performance of our proposed algorithm, it is compared with SCA [27] , WOA [12], HHA [28], SSA [29], BA [18], PSO [18] , and CSA [30]. The parameters of all compared algorithms are assigned based on those found in the published except the maximum iteration and population size for a fair comparison. For comparing our algorithm with all other algorithms fairly, we set the maximum iterations to 150 and the population size to 30 for all algorithms. Additionally, all algorithms run 20 independently times. Table 2 summarizes the values of the DWOA parameters.

Table 2 Parameter setting for the proposed EO

Parameter	Value
Number of runs	20
Population size	30
The maximum number of iteration	150
$a2$	2
$a1$	1

5.3 Solution Quality

In this section, the effectiveness of the proposed algorithm and other algorithms has been checked based on maximizing the Kapur's method for the test images until finding the optimized thresholding values of 2, 3, 4, 5, 10, 15, 20 thresholds levels. The threshold values and the corresponding fitness value obtained by our proposed algorithm on 7 test images are shown in Table 3. Fig.3 shows the segmented images obtained by the proposed algorithm.

Table 3

The results obtained by our proposed algorithm on the test images

Test Images	K	Threshold values	Fitness Value
Cameraman	2	125, 196	12.2844
	3	43, 101, 196	15.4002
	4	42, 96, 145, 196	18.5594
	5	24, 61, 99, 145, 196	21.3290
	10	20, 44, 69, 95, 119, 145, 169, 191, 210, 231	33.6083
	15	19, 34, 50, 66, 82, 97, 112, 128, 144, 159, 174, 190, 205, 222, 239	43.4237
	20	7,18,30,42,53,65,76,88,99,111,123,135,147,160,174,190,203,216,229,242	51.3972
House	2	95,208	10.7627
	3	47,97,208	13.6567
	4	25,61,98,208	16.2936
	5	64,122,163,202,209	18.6090
	10	19,46,71,95,121,148,175,203,210,230	31.1456
	15	12,25,45,61,76,95,112,125,140,155,170,186,203,210,230	40.8419
	20	12,25,36,46,60,72,85,97,112,125,139,153,166,179,191,203,209,221,234,247	48.9506
Lena	2	97,164	12.3447
	3	82,127,177	15.2123
	4	64,97,138,179	18.104
	5	63,94,127,162,194	20.6071
	10	41,59,78,97,119,140,160,179,197,216	31.4427
	15	41,57,71,84,96,110,123,137,150,164,178,191,204,217,230	40.2929
	20	35,45,54,63,74,87,98,109,121,133,144,155,165,175,184,194,205,216,226,235	47.66
lake	2	91,163	12.4920
	3	72,119,170	15.5467
	4	69,112,156,196	18.3288
	5	63,98,134,169,199	20.9908
	10	14,35,59,81,104,126,149,171,191,211	32.6219
	15	14,27,41,57,72,88,102,116,130,145,160,174,191,210,228	42.0987
	20	12,23,36,47,57,67,77,88,98,109,120,131,141,152,163,175,188,201,214,228	50.0027
JetPlane	2	69,172	12.2607
	3	68,125,181	15.5534
	4	64,104,144,184	18.3666
	5	58,89,123,155,186	20.9681
	10	34,51,68,87,107,127,147,167,186,205	31.9440
	15	26,39,52,65,78,91,105,119,132,145,159,172,185,198,212	40.7470
	20	19,29,42,56,65,75,85,94,105,116,127,138,149,160,171,182,192,202,213,225	48.1160
LivingRoom	2	91,170	12.6968
	3	46,103,175	15.9376
	4	46,98,148,196	18.9443
	5	44,89,129,168,201	21.7282
	10	23,45,67,89,111,135,159,182,204,235	33.8806
	15	16,31,47,63,79,96,112,128,144,160,175,191,208,226,239	43.7038

	20	10,20,32,42,55,66,77,89,100,112,124,136,148,160,173,186,200,213,226,239	51.7777
Peppers	2	73,145	12.5888
	3	60,112,163	15.6215
	4	50,98,138,178	18.4510
	5	41,75,112,150,191	21.1693
	10	23,43,60,78,98,117,136,156,176,196	32.4304
	15	20,34,47,61,75,88,101,115,130,144,158,172,186,200,215	41.3443
	20	16,31,42,52,61,71,81,91,101,110,120,130,140,149,159,170,181,191,203,215	48.6252

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(a) Cameraman segmented images

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(b) House segmented images

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

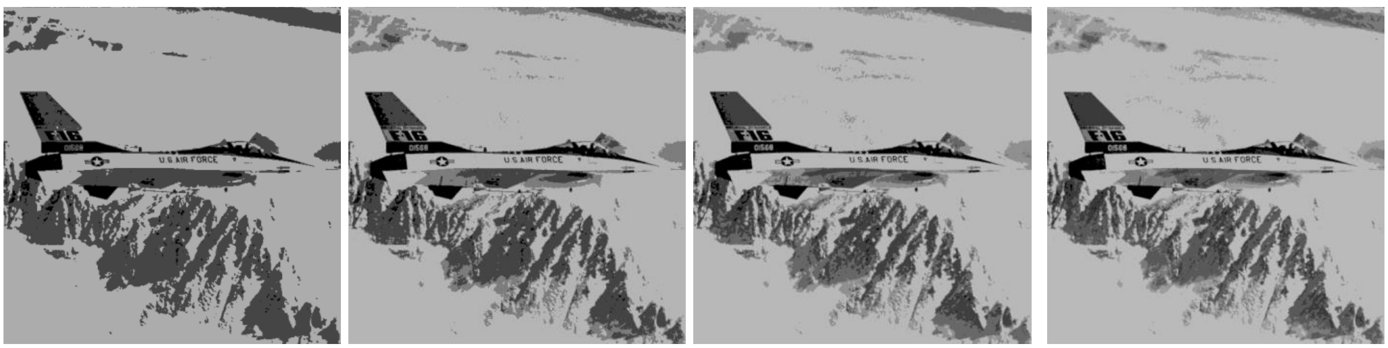
10-level thresholding 15-level thresholding 20-level thresholding

(c) Lena segmented images

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(d) Lake segmented images



2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(e) Jetplane segmented images

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(f) LivingRoom segmented images

2-level thresholding 3-level thresholding 4-level thresholding 5-level thresholding

10-level thresholding 15-level thresholding 20-level thresholding

(g) Peppers segmented images

Fig.3 Segmented images produced by the proposed algorithm.

5.5 Performance evaluation

For evaluating the performance of the algorithms, we have used five metrics to check the quality of the output images and have used another one to check the speedup of the algorithms. Those metrics are peak signal to noise ratio (PSNR), structured similarity index metric (SSIM), fitness value, signal to noise ratio (SNR), mean absolute error (MAE), and CPU time. The mathematical model of those metrics is:

1) Peak signal to noise ratio (PSNR) [31]: PSNR is used to measure the quality of the segmented images based on measuring the ratio between the square of the maximum pixel value (255) of a signal and the mean squared error which influences the quality of the segmented images and is calculated using the following formula:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{19}$$

Where MSE is the mean squared error and is calculated as follows:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}|A(i,j) - S(i,j)|}{M*N} \tag{20}$$

Where $A(i,j)$, $S(i,j)$ represent the original and segmented images, respectively. Note that the greater value of PSNR refers to the best performance.

2) Signal to noise ratio (SNR) [32]: SNR is used to measure the quality of the segmented images based on measuring the ratio between the squared average of the intensity values of the input image and the squared error between the original and segmented image which influences the quality of the segmented images and is calculated using the following formula:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{AI^2}{\mathrm{SE}^2}\right) \tag{21}$$

Where AI is the average of the intensity values of the original image and is calculated as follows:

$$\mathrm{AI} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} A(i,j)}{M*N} \tag{22}$$

And SE is the squared error and is calculated as follows:

$$\mathrm{SE} = \sum_{i=1}^{M}\sum_{j=1}^{N}|A(i,j) - S(i,j)| \tag{23}$$

Where $A(i,j)$, $S(i,j)$ represent the original and segmented images, respectively. Note that the greater value of SNR refers to the best performance. This metric is used to calculate how strong the noise corrupted the original image, unlike PSNR, that focuses on the high-intensity regions of the image. Therefore, SNR is good for images where intensity is equally distributed, while PSNR is good for those images where it varies a lot.

3) Maximum absolute error (MAE) [33]: MAE is the maximum absolute error between the original and segmented image and is calculated using the following formula:

$$\mathrm{MAE} = \max(|A(:) - S(:)|) \tag{24}$$

Where $A(:), and\ S(:)$ represents the original and segmented images, respectively. Note that the smaller value of MAE refers to the best performance.

All the previous metrics are traditional methods because they pay attention to finding the ratio of the error between the original and the segmented images and don't focus on the structure of the image after the segmentation process. As a result, the next metric, SSIM, is used to pay attention to the structure of the segmented image where it includes three factors: loss of correlation, luminance distortion, and contrast distortion between the original and segmented images.

4) Structured similarity index metric (SSIM) [31]: SSIM is a perceptual metric that qualifies the segmented image quality degradation using a variety of known properties of the human visual system. The mathematical model of SSIM is as follows:

$$\mathrm{SSIM}(\mathrm{O}, \mathrm{S}) = \frac{(2\,\mu_o\mu_s + a)(2\sigma_{os} + b)}{(\mu_o{}^2 + \mu_s{}^2 + a)(\sigma_o{}^2 + \sigma_s{}^2 + b)} \tag{25}$$

Where μ_o, μ_s represent the mean intensity of the original and segmented image; σ_o and σ_s symbolize the standard deviation of the original and segmented image; σ_{os} refers to the co-variance of the original and segmented image; and $a, and\ b$ is constant values and equal to 0.001 and 0.003 respectively. Please note that the greater value of SSIM can represent the best performance.

5) The average of the fitness function: this metric used to calculate the stability of the algorithms and is defined as:

$$\mathrm{F}_{\mathrm{avg}} = \frac{\sum_{i=0}^{T} F_i}{T} \tag{26}$$

Where T refers to the number of runs, and F_i refers to the best value.

6) The standard deviation: is also used to validate the steadiness of the results obtained by each algorithm within a number of runs, and is defined as follows:

$$S = \sqrt{\frac{\sum_{i=0}^{T} (F_i - F_{avg})^2}{T-1}} \tag{27}$$

Where T refers to the number of runs, F_i represents the fitness value obtained each run, and F_{avg} is the average of the fitness values in all runs.

7) The time complexity: is used to check the speedup of each algorithm for solving the multi-thresholding problem.

Table 4-10 shows the results of four metrics used for evaluating the performance of the algorithms. Based on the results introduced in those tables, the performance of all algorithms is roughly convergent on the small thresholds levels. With the increase of the thresholds level, the superiority of our proposed algorithms over all other algorithms are shown significantly, where our algorithm could roughly outperform on all other algorithms for PSNR, SSIM, MAE, SNR, and fitness value on the test images with the large threshold levels. Unfortunately, with the large thresholds level, the CPU time and standard deviation (Std) values increase a little compared with some other algorithms.

Table 4. Comparison of the performance of the algorithms on the cameraman test image

Test Images	K	Algorithms	F_{avg}	S	Time	PSNR	SSIM	MAE	SNR
Cameraman	2	EO	**12.2844**	**0**	**0.1434**	13.7969	0.7925	41.2870	1.8940
		WOA [12]	**12.2844**	**0.0000**	0.1690	13.7969	0.7917	41.2870	1.8940
		HHA [28]	12.2842	0.0005	0.1794	13.8141	0.7922	41.2870	1.8940
		SCA [27]	12.2838	0.0006	0.1690	**13.8250**	**0.7926**	**41.2044**	**1.9005**
		PSO [18]	12.2844	0.0000	0.1919	13.0533	0.7465	41.2870	1.8940
		BA [18]	12.2837	0.0012	0.1768	13.8502	**0.7938**	41.2139	1.8998
		CSA [30]	12.2840	0.0004	0.1721	13.8069	0.7906	41.2486	1.8971
		SSA [29]	**12.2844**	0.0000	0.3650	13.8012	0.7919	41.2870	1.8940
	3	EO	**15.4002**	**0.0000**	**0. 2876**	**14.3338**	**0.8139**	42.0935	**1.8948**
		WOA [12]	**15.4002**	**0.0000**	0.3078	14.3286	0.8073	42.0935	1.7926
		HHA [28]	15.3930	0.0076	0.3198	14.5207	0.8156	42.0905	1.7928
		SCA [27]	15.3947	0.0033	0.3089	14.4852	0.8139	41.2940	1.8510
		PSO [18]	15.4002	0.0000	0.3588	15.3131	0.8382	42.0935	1.7926
		BA [18]	15.3924	0.0071	0.3198	14.3130	0.8058	42.1300	1.7903
		CSA [30]	15.3938	0.0053	0.3125	14.4605	0.8123	**40.7181**	1.8940
		SSA [29]	15.3942	0.0228	0.5143	14.5558	0.8139	42.0947	1.7925
	4	EO	**18.5594**	**0**	**0. 4306**	**20.1657**	**0.9559**	21.7644	4.4901
		WOA [12]	18.5591	0.0005	0.4555	20.1560	0.9558	21.7592	4.4911
		HHA [28]	18.5291	0.0141	0.4571	20.1156	0.9555	**21.7362**	**4.4971**
		SCA [27]	18.5293	0.0164	0.4534	19.9939	0.9543	21.8126	4.4606
		PSO [18]	18.5594	0.0000	0.5221	17.5979	0.9027	21.8053	4.4811
		BA [18]	18.5057	0.0417	0.4612	19.6064	0.9498	22.9991	4.2218
		CSA [30]	18.5370	0.0145	0.4586	19.9669	0.9538	22.0028	4.4264
		SSA [29]	18.5565	0.0019	0.6656	20.0987	0.9557	21.8000	4.4826
	5	EO	**21.3264**	**0.001**	**0.5808**	**20.733**	**0.9618**	**20.2812**	**4.8617**
		WOA [12]	21.3258	0.0072	0.5949	20.7224	**0.9618**	20.6664	4.7569
		HHA [28]	21.2677	0.0229	0.6022	20.7053	**0.9618**	20.2847	4.8469
		SCA [27]	21.2613	0.0293	0.5990	20.9117	**0.9618**	20.2969	4.8396
		PSO [18]	21.3098	0.0044	0.7041	19.2672	0.9360	20.2986	4.8472
		BA [18]	21.2804	0.0337	0.6058	20.2055	0.9555	21.4206	4.5640
		CSA [30]	21.2750	0.0400	0.6032	20.5301	0.9586	20.7463	4.7171
		SSA [29]	21.3118	0.0152	0.8206	20.4344	0.9579	20.5886	4.7724
	10	EO	**33.559**	0.050	**0.711**	**26.6718**	**0.9893**	**11.4126**	**9.3626**
		WOA [12]	33.5526	**0.0334**	0.7472	25.4449	0.9859	11.7937	9.0135

	HHA [28]	33.0439	0.1693	0.7587	24.8088	0.9833	11.6119	9.1669
	SCA [27]	33.0859	0.1200	0.7582	24.9212	0.9836	12.0779	8.6372
	PSO [18]	33.5461	0.0000	0.8845	23.0699	0.9695	12.4835	8.3927
	BA [18]	33.1896	0.2011	0.7644	24.5267	0.9816	13.0362	8.0244
	CSA [30]	33.0688	0.1408	0.7530	25.2706	0.9849	12.5719	8.3527
	SSA [29]	33.4725	0.0861	0.9833	25.1925	0.9849	12.2799	8.6066
15	EO	**43.244**	**0.156**	**0.9251**	**29.486**	**0.9923**	**8.1601**	**12.9958**
	WOA [12]	43.2291	0.1125	0.9053	27.9440	0.9910	8.1766	12.8702
	HHA [28]	41.8325	0.3478	0.9225	27.1174	0.9886	8.8879	11.8311
	SCA [27]	42.0819	0.3419	0.9209	27.4277	0.9897	8.9703	11.4190
	PSO [18]	42.8887	0.0000	1.0764	25.8320	0.9826	8.8378	11.5463
	BA [18]	42.5354	0.2855	0.9240	27.7597	0.9902	8.9655	11.6672
	CSA [30]	42.0200	0.2837	0.9131	27.0290	0.9888	9.0170	11.3828
	SSA [29]	42.5775	0.2626	1.1476	27.5785	0.9899	8.5016	12.5065
20	EO	**51.15**	0.198	**1.0672**	**30.8999**	**0.9933**	6.4283	15.8243
	WOA [12]	51.0529	**0.1780**	1.0832	29.4731	0.9927	6.5941	15.3317
	HHA [28]	48.9322	0.3610	1.0884	28.8455	0.9913	7.1860	14.1617
	SCA [27]	49.3253	0.3700	1.0884	28.8850	0.9916	7.0786	14.1432
	PSO [18]	50.5297	0.2987	1.2782	27.4049	0.9877	7.3580	13.5498
	BA [18]	50.0055	0.5143	1.0998	29.9972	0.9929	6.3248	15.8404
	CSA [30]	49.0690	0.4144	1.0811	28.6733	0.9911	7.0554	13.9127
	SSA [29]	50.1877	0.5477	1.3172	29.9995	0.9929	**6.2552**	**16.0688**
30	EO	**63.3082**	0.5673	1.2439	**32.6225**	**0.9949**	4.4644	20.7363
	WOA [12]	63.1370	**0.5125**	1.2730	32.5757	**0.9949**	4.2421	21.8301
	HHA [28]	62.2145	0.5917	2.1835	31.9714	0.9945	4.8924	19.0855
	SCA [27]	59.2768	0.7013	**1.2412**	31.1996	0.9937	5.3065	17.6193
	PSO [18]	59.4927	1.0744	1.6957	31.6460	0.9941	5.0875	18.0668
	BA [18]	61.2218	1.0715	1.3213	32.4002	0.9946	3.9290	22.6048
	CSA [30]	59.1808	0.5654	1.3483	31.5213	0.9939	4.9423	18.7029
	SSA [29]	62.0128	0.8996	1.6897	33.0720	0.9951	**4.0405**	**22.6041**
40	EO	**71.9235**	**0.7027**	1.4649	**34.7056**	**0.9959**	3.1792	26.1123
	WOA [12]	70.8257	0.7329	1.4789	33.9084	0.9956	3.2867	25.5106
	HHA [28]	69.8032	0.7997	2.7732	33.7762	0.9954	3.3117	25.2440
	SCA [27]	66.1775	0.9773	**1.4492**	32.4990	0.9946	3.8227	22.6100
	PSO [18]	66.0217	1.2598	1.9198	33.6771	0.9954	3.9318	22.1119
	BA [18]	69.3100	1.1527	1.5314	34.0300	0.9955	3.0897	26.4990
	CSA [30]	66.0903	0.5959	1.5496	32.7482	0.9948	3.8766	21.9612
	SSA [29]	69.8982	0.8143	1.9070	34.5267	0.9958	3.0501	26.7834
50	EO	**76.9196**	1.4709	**1.7129**	**35.4517**	**0.9961**	**2.4548**	**30.1506**
	WOA [12]	75.6398	1.1764	1.7394	34.8725	0.9959	2.6661	28.9291
	HHA [28]	75.3161	0.7365	3.5308	34.8147	0.9959	2.6589	28.7246
	SCA [27]	71.2321	1.1958	1.6817	34.1234	0.9955	3.2313	24.7442
	PSO [18]	70.6307	1.4728	2.1637	34.0828	0.9955	3.0147	26.1857
	BA [18]	74.3661	1.1129	1.7613	34.7612	0.9958	2.6458	28.6713
	CSA [30]	70.8393	**0.4753**	1.7675	34.3572	0.9956	3.1115	25.4532

SSA [29]	74.8254	1.1156	2.1416	35.1455	0.9960	2.4567	29.9874

Bold values refer to the best results

Table 5. Comparison of the performance of the algorithms on the house test image

Test Images	K	Algorithms	F_{avg}	S	Time	PSNR	SSIM	MAE	SNR
House	2	EO	**10.7627**	**0**	**0.119**	**11.2157**	0.5425	**55.2293**	1.2857
		WOA [12]	10.7608	0.0085	0.1248	11.0424	0.5592	56.7420	**1.2884**
		HHA [28]	10.7613	0.0021	0.1238	11.1167	0.5697	56.7420	**1.2884**
		SCA [27]	10.7607	0.0025	0.1284	11.1216	0.5709	57.0408	1.2773
		PSO [18]	**10.7627**	0.0000	0.1258	11.1387	**0.5711**	56.7420	1.2884
		BA [18]	10.7531	0.0136	0.1290	10.8518	0.5379	57.2415	1.2697
		CSA [30]	10.7605	0.0023	0.1248	11.1117	0.5696	56.8646	1.2840
		SSA [29]	10.7608	0.0085	0.1269	11.0402	0.5590	57.5725	1.2640
	3	EO	**13.6567**	**0**	**0.2506**	**11.9354**	**0.5225**	47.9807	1.4604
		WOA [12]	13.6530	0.0106	0.2522	11.9263	0.5221	47.8769	1.4670
		HHA [28]	13.6129	0.0265	0.2506	11.8791	0.5210	**47.8497**	**1.4692**
		SCA [27]	13.5975	0.0312	0.2590	11.8360	0.5207	49.0944	1.4264
		PSO [18]	**13.6567**	**0.0000**	0.2553	**11.9354**	**0.5225**	48.2700	1.4487
		BA [18]	13.5599	0.0479	0.2595	11.6388	0.5097	50.6084	1.3718
		CSA [30]	13.5937	0.0353	0.2600	11.8155	0.5183	48.5743	1.4397
		SSA [29]	13.6112	0.0529	0.2584	11.8135	0.5205	47.9266	1.4643
	4	EO	16.0188	0.2925	**0.3721**	**15.0743**	**0.7218**	**42.6569**	**1.9717**
		WOA [12]	16.2098	0.1852	0.3827	12.6672	0.5476	46.1519	1.4986
		HHA [28]	16.1321	0.0901	0.3848	11.8080	0.4876	46.7417	1.4782
		SCA [27]	16.1486	0.1128	0.3890	12.0096	0.5072	48.1038	1.4476
		PSO [18]	**16.2619**	**0.1340**	0.3874	12.3917	0.5268	47.3385	1.4618
		BA [18]	16.1003	0.0877	0.3900	11.6219	0.4770	49.4178	1.3875
		CSA [30]	16.1053	0.1196	0.3900	11.8304	0.4944	47.8255	1.4430
		SSA [29]	16.1485	0.1940	0.3947	12.4944	0.5464	46.3411	1.4929
	5	EO	18.5264	0.0838	**0.5025**	**20.1606**	**0.9532**	**19.7088**	**5.7774**
		WOA [12]	**18.5586**	**0.0802**	0.5143	17.5565	0.8504	23.8221	4.7673
		HHA [28]	18.4284	0.0576	0.5158	18.4355	0.8992	27.5225	4.1619
		SCA [27]	18.4627	0.0758	0.5221	17.6278	0.8735	26.8551	3.9946
		PSO [18]	18.5537	0.0782	0.5195	19.2000	0.9242	24.4528	4.4120
		BA [18]	18.4104	0.0942	0.5216	17.4805	0.8533	26.1786	4.4082
		CSA [30]	18.4397	0.0720	0.5216	18.0583	0.9109	25.0154	4.4971
		SSA [29]	18.4821	0.0764	0.5309	18.6386	0.8816	20.0484	5.6614
	10	EO	**31.0564**	**0.1287**	**0.6081**	**26.9759**	**0.9883**	**8.8219**	**12.2788**
		WOA [12]	30.8506	0.2167	0.6589	26.7438	0.9874	9.3982	11.8711
		HHA [28]	30.1960	0.2082	0.6536	25.5445	0.9849	9.4467	11.5241
		SCA [27]	30.1948	0.2079	0.6692	25.8634	0.9857	10.6415	10.4215
		PSO [18]	30.8606	0.1610	0.6625	26.8022	0.9880	10.5015	10.8293
		BA [18]	30.1908	0.2884	0.6677	25.1112	0.9861	12.3450	9.7208
		CSA [30]	30.3261	0.2001	0.6614	25.7182	0.9854	11.5139	9.8193
		SSA [29]	30.5845	0.2853	0.6713	26.3585	0.9877	9.9999	11.4091

15	EO	**40.6614**	**0.1430**	**0.9714**	**30.5336**	**0.9938**	**6.0371**	**18.5776**
	WOA [12]	40.3845	0.1654	0.8091	29.7265	0.9926	6.2002	17.9594
	HHA [28]	39.2259	0.2674	0.8029	28.4181	0.9912	6.7305	17.2143
	SCA [27]	39.3150	0.3221	0.8200	28.4358	0.9910	7.9400	14.3308
	PSO [18]	39.9169	0.5850	0.8154	29.8850	0.9933	7.5789	15.0034
	BA [18]	39.4323	0.3175	0.8211	27.3268	0.9897	9.9844	12.1911
	CSA [30]	39.2596	0.2846	0.8143	28.6210	0.9914	8.0700	14.6413
	SSA [29]	39.9713	0.3312	0.8268	28.9015	0.9925	7.7045	15.4717
20	EO	**48.7683**	**0.2023**	**0.96125**	**32.7884**	**0.9954**	**4.5342**	**25.1879**
	WOA [12]	48.3588	0.3438	0.9724	31.8537	0.9947	5.2937	22.0344
	HHA [28]	46.3133	0.3709	0.9578	30.1709	0.9935	5.6335	20.8722
	SCA [27]	46.4282	0.5768	0.9812	29.5841	0.9919	5.6631	19.9296
	PSO [18]	47.0462	0.6336	0.9823	31.3374	0.9943	6.7212	17.4141
	BA [18]	47.1150	0.5782	0.9823	29.9211	0.9938	6.8967	17.8015
	CSA [30]	46.5362	0.4814	0.9750	30.1527	0.9928	6.1735	18.5783
	SSA [29]	47.4646	0.5169	0.9843	30.3892	0.9936	5.6881	21.1915
30	EO	**61.1904**	**0.3413**	**1.1768**	**36.2045**	**0.9970**	**3.0851**	**36.7416**
	WOA [12]	60.3357	0.8377	1.2610	34.1053	0.9960	3.7865	31.1914
	HHA [28]	59.6055	0.7709	2.0759	33.9871	0.9960	3.5547	31.6996
	SCA [27]	56.9888	0.8687	1.1945	33.4945	0.9954	4.4075	25.6984
	PSO [18]	56.6042	1.4351	1.1981	33.2089	0.9956	4.3022	25.7520
	BA [18]	58.8423	0.6240	1.1627	33.2784	0.9957	5.1013	24.2985
	CSA [30]	57.1936	0.8620	1.3821	33.3111	0.9952	4.3774	26.1660
	SSA [29]	59.1837	0.6519	1.2085	32.9402	0.9955	4.4051	27.3036
40	EO	**70.0402**	**0.7249**	**1.3868**	**38.0745**	**0.9972**	**2.4244**	**47.0791**
	WOA [12]	68.5505	0.9742	1.4622	36.8750	0.9969	2.8058	39.2284
	HHA [28]	67.4850	0.8561	2.6505	36.0359	0.9967	2.9605	38.0215
	SCA [27]	63.7988	0.8734	1.3968	34.7637	0.9961	3.8461	29.1458
	PSO [18]	63.8004	1.2725	1.4134	35.7031	0.9964	3.4486	31.4232
	BA [18]	67.0638	1.1561	1.3645	34.5076	0.9958	3.2962	36.1988
	CSA [30]	63.7188	0.5633	1.5792	34.4196	0.9957	3.4681	32.1761
	SSA [29]	67.3441	1.0170	1.4030	35.4812	0.9965	3.3302	35.8302
50	EO	**75.9533**	1.4318	**1.6347**	**39.7232**	**0.9974**	**1.9993**	**54.6155**
	WOA [12]	73.0770	**1.2156**	1.6827	38.2366	0.9972	2.4069	46.5007
	HHA [28]	73.3144	1.4153	3.4029	38.0962	0.9972	2.5117	44.6755
	SCA [27]	68.7655	0.9520	1.6172	36.0854	0.9967	2.7178	39.6693
	PSO [18]	68.7110	1.4802	1.6495	37.5117	0.9969	2.9352	38.0673
	BA [18]	72.0987	1.4241	1.5912	37.2455	0.9969	2.6987	41.9211
	CSA [30]	69.0858	0.8755	1.7908	36.8188	0.9968	2.8370	38.6928
	SSA [29]	72.4336	1.6134	1.6229	36.9563	0.9969	2.4930	44.2793

Bold values refer to the best results

Table 6. Comparison of the performance of the algorithms on the Lena test image

Test Images	K	Algorithm	F_{avg}	S	Time	PSNR	SSIM	MAE	SNR
	2	EO	**12.3447**	**0.0000**	**0.1290**	**14.6273**	**0.8295**	41.3444	2.1283

Lena		WOA [12]	**12.3447**	**0.0000**	0.1331	**14.6273**	**0.8295**	41.3444	2.1283
		HHA [28]	12.3444	0.0005	0.1471	14.6049	0.8290	41.3631	2.1272
		SCA [27]	12.3445	0.0003	0.1352	14.6033	0.8289	41.3843	2.1260
		PSO [18]	12.3447	0.0000	0.1336	14.6273	0.8295	41.3444	2.1283
		BA [18]	12.3443	0.0010	0.1451	14.6354	0.8296	**41.2583**	**2.1336**
		CSA [30]	12.3444	0.0003	0.1311	14.6051	0.8291	41.4102	2.1244
		SSA [29]	12.3447	0.0000	0.1601	14.6273	0.8295	41.3444	2.1283
	3	EO	**15.3123**	**0.0000**	**0.2598**	**17.2096**	**0.9004**	29.8377	3.1337
		WOA [12]	**15.3123**	**0.0000**	0.2657	**17.2096**	**0.9004**	29.8377	3.1337
		HHA [28]	15.3097	0.0022	0.2808	17.1748	0.8992	29.8284	3.1345
		SCA [27]	15.3108	0.0019	0.2819	17.2169	0.9001	29.8457	**3.1380**
		PSO [18]	15.3123	0.0000	0.2668	**17.2096**	**0.9004**	29.8377	3.1337
		BA [18]	15.3081	0.0040	0.2792	17.3078	0.9022	29.6192	3.1594
		CSA [30]	15.3108	0.0020	0.2689	17.2136	0.9002	29.8500	3.1303
		SSA [29]	15.3122	0.0001	0.2933	17.2357	0.9006	**29.8193**	3.1354
	4	EO	18.0000	0.0122	0.4196	18.6635	0.9264	24.8042	3.8327
		WOA [12]	**18.0104**	**0.0001**	**0.4342**	**19.0336**	**0.9302**	**23.9498**	**3.9895**
		HHA [28]	17.9846	0.0102	0.4160	18.8434	0.9261	24.1276	3.9568
		SCA [27]	17.9913	0.0110	0.4165	18.8700	0.9269	24.3818	3.9180
		PSO [18]	18.0019	0.0119	0.4030	18.8030	0.9262	24.3850	3.9091
		BA [18]	17.9923	0.0160	0.4144	18.9344	0.9281	24.3986	3.9099
		CSA [30]	17.9936	0.0109	0.4269	18.8564	0.9269	24.4357	3.8935
		SSA [29]	18.0045	0.0089	0.4305	18.9439	0.9288	24.2162	3.9429
	5	EO	**20.6071**	**0.0001**	**0.5522**	**19.8624**	**0.9386**	20.9367	4.5253
		WOA [12]	20.6069	0.0003	0.5751	19.8452	0.9384	20.9511	4.5189
		HHA [28]	20.5342	0.0295	0.5538	19.7922	0.9367	20.9772	4.5118
		SCA [27]	20.5623	0.0299	0.5528	19.7358	0.9363	21.0487	4.5107
		PSO [18]	20.6071	0.0000	0.5444	19.8736	0.9388	20.9575	4.5250
		BA [18]	20.5186	0.0739	0.5569	20.2111	0.9422	20.1914	4.6977
		CSA [30]	20.5664	0.0186	0.5668	19.7691	0.9366	21.0498	4.5011
		SSA [29]	20.6009	0.0053	0.5704	19.9327	0.9392	**20.9021**	**4.5349**
	10	EO	**31.4215**	0.0325	0.7545	26.5233	0.9837	9.8728	10.317
		WOA [12]	31.4174	**0.0198**	0.7197	26.1357	0.9817	10.8347	9.1532
		HHA [28]	30.8171	0.1787	0.6984	25.4852	0.9780	10.0709	10.043
		SCA [27]	31.0681	0.1220	**0.6979**	24.9925	0.9747	11.3343	8.7208
		PSO [18]	31.3751	0.0802	0.7316	25.6028	0.9773	11.8810	7.9884
		BA [18]	30.9608	0.2426	0.7056	**26.9516**	**0.9857**	10.2541	10.094
		CSA [30]	31.0157	0.1508	0.7135	24.6378	0.9720	11.8237	8.2315
		SSA [29]	31.2548	0.1250	0.7155	26.6497	0.9836	**9.4234**	**10.880**
	15	EO	**40.1678**	**0.1115**	0.8611	**30.1091**	**0.9917**	6.4413	16.173
		WOA [12]	40.0730	0.1209	0.8778	29.5605	0.9908	6.9240	14.676
		HHA [28]	38.7786	0.2637	0.8559	27.4818	0.9842	7.1287	14.213
		SCA [27]	39.0505	0.4160	**0.8528**	28.2797	0.9871	7.9559	12.347
		PSO [18]	40.0736	0.1307	0.8897	29.9382	0.9910	7.6676	12.719
		BA [18]	39.4162	0.3614	0.8658	29.9648	0.9915	6.5639	16.125
		CSA [30]	39.0500	0.4209	0.8700	27.8289	0.9854	8.1588	12.249

		SSA [29]	39.8302	0.1827	0.8715	30.1063	0.9915	**6.3839**	**16.430**
	20	EO	**47.4679**	**0.1087**	1.0291	32.5902	**0.9947**	**4.8769**	**21.399**
		WOA [12]	47.3220	0.1203	1.0432	32.0638	0.9938	5.1969	20.102
		HHA [28]	44.9852	0.5899	1.0187	29.3923	0.9888	5.4039	18.996
		SCA [27]	45.2668	0.7458	**1.0166**	30.1228	0.9905	6.0443	16.554
		PSO [18]	47.0159	0.5126	1.0634	32.3088	0.9940	5.9689	16.586
		BA [18]	45.6805	0.9362	1.0359	31.5908	0.9931	5.5031	19.225
		CSA [30]	45.2688	0.5634	1.0400	29.7931	0.9889	6.2080	16.121
		SSA [29]	46.2318	0.6983	1.0348	**32.6244**	0.9944	4.9047	21.426
	30	EO	**58.6289**	**0.2896**	1.2324	**36.2744**	**0.9962**	**3.1950**	**32.706**
		WOA [12]	57.6599	0.7104	1.2033	34.7044	0.9954	3.7773	27.096
		HHA [28]	56.8433	0.6724	2.0743	34.1934	0.9950	4.0877	24.315
		SCA [27]	54.1149	1.0301	1.2215	32.8770	0.9938	4.6811	20.434
		PSO [18]	56.4881	1.2084	1.2319	35.1079	0.9954	4.1470	23.508
		BA [18]	55.0831	0.9709	1.2054	34.4217	0.9952	3.7537	27.215
		CSA [30]	53.6043	0.5445	1.2880	32.4283	0.9933	4.2538	23.330
		SSA [29]	56.1225	1.2295	**1.2189**	35.0641	0.9956	3.3228	31.284
	40	EO	**65.8907**	0.6009	1.4544	**38.9034**	**0.9969**	**2.3766**	**42.999**
		WOA [12]	64.0561	**0.5831**	1.4097	36.6790	0.9964	2.7634	35.764
		HHA [28]	62.6129	1.1788	2.6536	35.6201	0.9956	2.9015	34.457
		SCA [27]	59.2393	0.9832	1.4290	34.9160	0.9952	3.1834	30.506
		PSO [18]	61.9209	1.5461	1.4472	37.0017	0.9964	3.2715	29.671
		BA [18]	61.3615	1.3195	1.4134	36.4300	0.9962	2.9663	33.826
		CSA [30]	59.3639	0.5898	1.4939	35.0198	0.9953	3.4487	27.835
		SSA [29]	61.9115	1.2080	1.4258	36.7591	0.9963	2.6371	38.294
	50	EO	**70.1318**	**0.9632**	1.6994	**40.2650**	**0.9973**	**1.9043**	**52.944**
		WOA [12]	67.9429	1.2033	1.6349	37.9935	0.9966	2.2966	42.919
		HHA [28]	66.7760	1.1899	3.4039	37.9951	0.9966	2.4220	39.549
		SCA [27]	62.8502	0.7897	1.6552	36.7777	0.9962	2.6218	36.540
		PSO [18]	65.2505	1.8632	1.6838	38.3370	0.9967	2.5975	36.839
		BA [18]	66.0501	1.6181	1.6489	37.9150	0.9966	2.5653	38.036
		CSA [30]	62.9774	0.6415	1.7212	36.9014	0.9963	2.6818	35.851
		SSA [29]	66.2675	1.2287	1.6578	38.3963	0.9967	2.2091	45.147

Bold values refer to the best results

Table 7. Comparison of the performance of the algorithms on the lake test image

Test Images	K	Algorithms	F_{avg}	S	Time	PSNR	SSIM	MAE	SNR
Lake	2	EO	**12.4920**	**0.0000**	**0.1342**	14.7917	0.8751	41.3834	2.3869
		WOA [12]	**12.4920**	**0.0000**	0.1440	14.7917	0.8751	41.3834	2.3869
		HHA [28]	12.4919	0.0001	0.1440	14.7950	0.8767	41.3834	2.3869
		SCA [27]	12.4919	0.0001	0.1435	14.7892	0.8766	**41.3244**	**2.3907**
		PSO [18]	**12.4920**	**0.0000**	0.1388	14.7917	0.8751	41.3834	2.3869
		BA [18]	12.4916	0.0006	0.1388	14.7881	0.8757	41.4097	2.3848
		CSA [30]	12.4918	0.0003	0.1404	**14.7965**	**0.8767**	41.3454	2.3894
		SSA [29]	**12.4920**	**0.0000**	0.1409	13.3541	0.7438	41.3834	2.3869

3	EO	**15.5467**	**0.0000**	**0.2715**	16.5765	**0.9169**	32.9230	3.0871
	WOA [12]	**15.5467**	**0.0000**	**0.2907**	**16.5771**	**0.9169**	32.9263	3.0871
	HHA [28]	15.5427	0.0029	0.2860	16.5408	0.9159	32.9263	3.0871
	SCA [27]	15.5442	0.0030	0.2860	16.5641	0.9164	**32.8488**	**3.0953**
	PSO [18]	15.5467	0.0000	0.2849	16.5765	0.9169	32.9230	3.0871
	BA [18]	15.5307	0.0166	0.2850	16.5681	0.9164	32.8721	3.0891
	CSA [30]	15.5433	0.0036	0.2860	16.5532	0.9163	33.2307	3.0562
	SSA [29]	15.5466	0.0005	0.2834	16.5749	0.9169	32.9230	3.0871
4	EO	18.3224	0.0152	**0.4165**	**17.5216**	**0.9329**	29.4365	3.5535
	WOA [12]	**18.3287**	**0.0003**	0.4358	17.4997	0.9329	29.5936	3.5455
	HHA [28]	18.3016	0.0135	0.4285	17.4193	0.9319	29.5701	3.5483
	SCA [27]	18.3171	0.0085	0.4311	17.5207	0.9339	29.5655	3.5464
	PSO [18]	18.3288	0.0000	0.4285	17.5020	0.9329	29.6201	3.5420
	BA [18]	18.3041	0.0266	0.4295	17.6490	0.9352	**29.1009**	**3.5893**
	CSA [30]	18.3152	0.0102	0.4295	17.5008	0.9038	29.5489	3.5486
	SSA [29]	18.3136	0.0199	0.4264	17.5088	0.9033	29.5187	3.5487
5	EO	20.9908	**0.0001**	**0.5590**	18.7270	0.9462	24.4387	4.2417
	WOA [12]	2**0.9939**	0.0299	0.5918	**18.8458**	**0.9475**	24.6161	4.2083
	HHA [28]	20.9214	0.0304	0.5782	18.8914	0.9491	24.5804	4.2171
	SCA [27]	20.9626	0.0133	0.5793	18.7197	0.9471	24.6260	4.2317
	PSO [18]	20.9908	0.0000	0.5761	18.7387	0.9463	24.6865	4.2035
	BA [18]	20.9134	0.0366	0.5798	19.2418	0.9527	**24.4111**	4.2877
	CSA [30]	20.9434	0.0203	0.5761	18.8293	0.9483	24.5475	**4.2543**
	SSA [29]	20.9847	0.0042	0.5710	18.8063	0.9481	24.4850	4.2503
10	EO	32.5814	0.1182	**0.7228**	26.3860	0.9893	**10.5008**	**10.8580**
	WOA [12]	**32.6029**	**0.0184**	0.7426	**26.4193**	**0.9894**	10.5582	10.7679
	HHA [28]	32.0070	0.1446	0.7342	25.2544	0.9859	10.7437	10.4896
	SCA [27]	32.1048	0.1371	0.7561	25.6002	0.9872	11.3729	9.6723
	PSO [18]	32.4701	0.1266	0.7337	25.9858	0.9884	11.4179	9.6156
	BA [18]	32.2548	0.1257	0.7550	25.2602	0.9862	11.7618	9.4053
	CSA [30]	32.0538	0.1492	0.7670	25.3692	0.9863	11.0588	9.8870
	SSA [29]	32.4047	0.1161	0.7862	25.9112	0.9881	10.9992	10.1899
15	EO	**42.0192**	**0.1043**	**0.8913**	**29.4342**	**0.9936**	**7.4037**	**15.4627**
	WOA [12]	41.8490	0.1826	0.9100	29.0251	0.9931	7.6992	14.7368
	HHA [28]	40.6283	0.2680	0.8975	27.6554	0.9907	7.7869	14.3437
	SCA [27]	40.9120	0.3595	0.9214	27.9462	0.9913	8.2441	13.2610
	PSO [18]	41.6167	0.3718	0.9049	29.1700	0.9933	8.4949	12.7127
	BA [18]	41.0659	0.4870	0.9256	27.9582	0.9912	8.0212	13.9387
	CSA [30]	40.9355	0.3878	0.9323	28.2737	0.9918	8.2484	13.0700
	SSA [29]	41.5014	0.3236	0.9532	28.5795	0.9924	7.8726	14.3211
20	EO	**49.7770**	0.2631	**1.0681**	**32.0049**	**0.9955**	**5.3679**	**21.3698**
	WOA [12]	49.6482	0.1743	1.0868	31.6511	0.9953	5.6575	20.0367
	HHA [28]	47.4934	0.3352	1.0743	29.7890	0.9934	5.7658	19.2902
	SCA [27]	47.5613	0.3858	1.0956	29.7542	0.9931	6.4262	16.6836
	PSO [18]	48.4916	0.6912	1.0843	30.5614	0.9941	6.7822	15.7422
	BA [18]	48.3781	0.5741	1.1040	30.4641	0.9941	6.3533	17.4825

	CSA [30]	47.5522	0.5255	1.1128	29.8052	0.9934	6.6947	15.9844
	SSA [29]	49.1386	0.2975	1.1294	31.1000	0.9949	5.8653	19.2431
30	EO	**61.5921**	**0.2990**	1.2870	**35.3189**	**0.9967**	**3.6667**	**30.5689**
	WOA [12]	61.1710	0.6199	1.2907	34.5663	0.9965	3.8949	28.3187
	HHA [28]	60.2210	0.7501	2.1294	33.9175	0.9962	4.0242	26.9063
	SCA [27]	57.4127	0.7239	1.2745	32.5771	0.9954	4.7171	22.2313
	PSO [18]	58.0381	1.1149	1.3203	33.0030	0.9957	4.7268	22.2799
	BA [18]	59.1389	0.9637	**1.2860**	33.4772	0.9960	4.2788	25.5267
	CSA [30]	57.2208	0.6304	1.4512	32.2977	0.9952	4.7078	22.4579
	SSA [29]	59.9185	0.8352	1.2865	33.9843	0.9963	3.6881	30.0006
40	EO	**69.5696**	0.8464	1.5169	**37.5886**	**0.9972**	**2.7420**	**40.0440**
	WOA [12]	68.3939	0.6245	1.5075	36.5759	0.9970	2.9689	36.1836
	HHA [28]	67.3841	1.1071	2.7082	36.3829	0.9970	3.0029	35.5285
	SCA [27]	63.8521	0.9471	**1.4893**	35.0020	0.9965	3.6089	28.9616
	PSO [18]	64.4272	1.3196	1.5486	35.5377	0.9967	3.6750	28.0254
	BA [18]	67.3117	1.0749	1.5065	35.9752	0.9968	3.1911	33.5231
	CSA [30]	63.7721	**0.6377**	1.6738	34.8239	0.9965	3.7095	27.9194
	SSA [29]	67.6693	0.9444	1.5075	36.6366	0.9970	2.8646	37.5418
50	EO	**74.9453**	1.2403	1.7722	**39.2284**	**0.9975**	**2.0903**	**51.6973**
	WOA [12]	72.9805	0.9874	1.7420	37.5554	0.9971	2.3819	44.4985
	HHA [28]	72.4774	0.8230	3.4617	37.8665	0.9972	2.5191	41.0487
	SCA [27]	68.2539	1.1940	**1.7207**	36.4376	0.9969	2.8593	35.8549
	PSO [18]	67.9253	1.5303	1.8003	36.2686	0.9968	3.2193	31.8963
	BA [18]	72.4789	1.4341	1.7550	37.7273	0.9972	2.5743	40.3612
	CSA [30]	68.2573	**0.8208**	1.9130	35.9505	0.9968	2.9031	35.0681
	SSA [29]	71.9562	1.3091	1.7352	37.7008	0.9972	2.3248	45.3675

Bold values refer to the best results

Table 8. Comparison the performance of the algorithms on the jetPlane test image

Test Images	K	Algorithm	F_{avg}	Std	Time	PSNR	SSIM	MAE	SNR
JetPlane	2	EO	**12.2607**	**0.0000**	**0.1310**	15.8661	0.8978	**36.6315**	**3.6446**
		WOA [12]	**12.2607**	**0.0000**	0.1347	15.8661	0.8978	**36.6315**	**3.6446**
		HHA [28]	12.2605	0.0003	0.1357	15.8459	0.8982	**36.6315**	**3.6446**
		SCA [27]	12.2604	0.0004	0.1433	15.8506	0.8983	36.8220	3.6279
		PSO [18]	12.2607	0.0000	0.1420	**15.8661**	0.8978	**36.6315**	**3.6446**
		BA [18]	12.2589	0.0028	0.1357	15.7267	**0.8989**	37.1103	3.6028
		CSA [30]	12.2602	0.0005	0.1415	15.8555	0.8974	36.7457	3.6373
		SSA [29]	12.2606	0.0001	0.1383	15.8633	0.8979	36.6315	3.6446
	3	EO	**15.5534**	**0.0000**	**0.2631**	18.8715	**0.9489**	26.1988	5.4702
		WOA [12]	**15.5534**	**0.0000**	0.2735	18.8715	**0.9489**	26.1762	5.4744
		HHA [28]	15.5460	0.0059	0.2730	18.8663	0.9488	26.1530	5.4786
		SCA [27]	15.5485	0.0040	0.2832	18.9296	0.9487	**26.0788**	**5.4886**
		PSO [18]	15.5534	0.0000	0.2824	18.8715	0.9489	26.1988	5.4702
		BA [18]	15.5361	0.0149	0.2777	18.5150	0.9476	27.3284	5.2521
		CSA [30]	15.5480	0.0051	0.2787	18.8478	**0.9489**	25.9140	5.5132
		SSA [29]	15.5531	0.0005	0.2766	18.8701	**0.9489**	26.1988	5.4702

4	EO	**18.3666**	**0.0000**	**0.4004**	**20.5346**	**0.9632**	21.5716	6.7817
	WOA [12]	18.3665	0.0002	0.4165	20.5333	0.9631	21.5654	6.7813
	HHA [28]	18.3483	0.0096	0.4150	20.5455	0.9625	**21.5570**	6.7819
	SCA [27]	18.3524	0.0093	0.4231	20.4705	0.9624	21.6961	6.7340
	PSO [18]	18.3666	0.0000	0.4254	20.5346	0.9632	21.5787	**6.7772**
	BA [18]	18.3394	0.0237	0.4191	20.0691	0.9599	23.2694	6.2904
	CSA [30]	18.3526	0.0092	0.4197	20.4613	0.9621	21.9565	6.6552
	SSA [29]	18.3646	0.0013	0.4274	20.4693	0.9626	21.6081	6.7681
5	EO	**20.9681**	**0.0001**	**0.5372**	**21.5758**	0.9688	19.0359	7.7458
	WOA [12]	20.9670	0.0015	0.5606	21.5755	**0.9690**	**19.0182**	**7.7501**
	HHA [28]	20.9043	0.0372	0.5590	21.5190	0.9681	19.0304	7.7437
	SCA [27]	20.9181	0.0281	0.5744	21.3830	0.9678	19.2447	7.6567
	PSO [18]	20.9677	0.0016	0.5699	21.5557	0.9686	19.0424	7.7378
	BA [18]	20.9071	0.0350	0.5658	20.9257	0.9654	21.1863	6.9620
	CSA [30]	20.9212	0.0296	0.5616	21.4450	0.9684	19.2519	7.6724
	SSA [29]	20.9617	0.0044	0.5689	21.4720	0.9683	19.2458	7.6613
10	EO	**31.9308**	**0.0101**	**0.6921**	27.4532	0.9888	**9.2549**	**16.1392**
	WOA [12]	31.9289	0.0116	0.7129	**27.4908**	**0.9889**	9.2687	16.0849
	HHA [28]	31.3379	0.1393	0.7082	26.2019	0.9843	9.2776	16.0656
	SCA [27]	31.5621	0.1238	0.7278	26.8751	0.9867	9.2790	15.8161
	PSO [18]	31.9256	0.0121	0.7233	27.4172	0.9888	10.6774	13.9859
	BA [18]	31.5200	0.2098	0.7202	24.5911	0.9785	13.3342	11.5022
	CSA [30]	31.5612	0.1610	0.7140	27.0364	0.9872	10.8508	13.8300
	SSA [29]	31.7915	0.1338	0.7212	26.1330	0.9853	10.6129	14.2227
15	EO	**40.6672**	**0.0757**	**0.8772**	**30.4589**	**0.9934**	6.7394	22.4170
	WOA [12]	40.5819	0.0767	0.8778	30.4045	**0.9934**	**6.5042**	**23.2433**
	HHA [28]	39.2368	0.3218	0.8668	28.4239	0.9890	6.9044	21.8527
	SCA [27]	39.5422	0.3147	0.8895	29.0816	0.9895	6.8542	21.3423
	PSO [18]	40.6508	0.0742	0.8892	30.2867	0.9931	7.4082	19.7725
	BA [18]	39.7686	0.3811	0.8830	26.8733	0.9850	10.6136	15.3102
	CSA [30]	39.6246	0.2212	0.8819	29.4846	0.9909	7.5575	19.1437
	SSA [29]	40.1488	0.3497	0.8897	28.1735	0.9891	8.2298	18.7036
20	EO	**47.8430**	**0.1741**	1.0618	**32.2753**	**0.9948**	**5.1080**	**30.0551**
	WOA [12]	47.7750	0.2211	1.0520	32.2716	0.9947	5.2473	28.8003
	HHA [28]	45.3242	0.4908	**1.0338**	30.5148	0.9918	5.5970	26.7803
	SCA [27]	45.5483	0.7395	1.0642	30.8613	0.9926	5.6038	25.8405
	PSO [18]	47.1357	0.5950	1.0676	31.2661	0.9932	6.2757	23.0224
	BA [18]	46.3258	0.9924	1.0598	29.4117	0.9902	7.1395	21.3653
	CSA [30]	45.5278	0.6125	1.0577	30.5598	0.9918	6.2882	22.8345
	SSA [29]	47.0391	0.5357	1.0603	29.9785	0.9908	6.4242	23.8412
30	EO	**58.9933**	**0.3335**	**1.2527**	**35.6940**	**0.9964**	**3.5851**	**42.1755**
	WOA [12]	58.0223	0.4569	1.2782	35.3259	0.9961	3.7214	39.2660
	HHA [28]	57.4042	1.0590	2.0743	34.5133	0.9955	3.8537	37.4981
	SCA [27]	54.5935	1.1665	1.2751	33.1971	0.9944	4.0513	34.6549
	PSO [18]	56.4425	1.1561	1.2828	33.7168	0.9947	4.7686	30.3058
	BA [18]	56.0881	1.1442	1.2678	33.1543	0.9945	4.5214	33.1172

		CSA [30]	54.0246	0.7477	1.3146	32.8699	0.9938	4.3836	31.9836
		SSA [29]	56.9536	0.9659	1.2881	34.5137	0.9955	4.5030	32.7732
	40	EO	**66.2906**	0.9599	1.4877	**37.4285**	**0.9968**	**2.7846**	**53.4728**
		WOA [12]	64.3415	1.2822	1.4903	37.1499	0.9966	2.8817	50.0711
		HHA [28]	64.0238	0.7589	2.6510	36.9181	0.9965	2.9402	48.3972
		SCA [27]	59.5460	0.8812	1.4898	35.9172	0.9961	2.9700	46.0341
		PSO [18]	61.9342	1.7053	1.5106	35.8282	0.9957	3.5422	39.1334
		BA [18]	63.0703	1.4799	**1.4872**	35.9545	0.9957	3.3583	43.6569
		CSA [30]	59.8482	**0.7919**	1.5231	34.9451	0.9954	3.2178	43.1621
		SSA [29]	63.5957	1.1464	1.5033	36.4755	0.9962	3.1298	47.1536
	50	EO	**70.5503**	0.7604	1.7446	**39.0207**	**0.9969**	**2.0657**	**70.0110**
		WOA [12]	68.4693	1.0990	1.7223	38.7043	0.9968	2.2555	61.0586
		HHA [28]	67.8117	1.4443	3.4029	37.9133	0.9966	2.4268	57.2823
		SCA [27]	64.2197	1.1649	**1.7280**	37.4836	0.9965	2.4932	55.2544
		PSO [18]	65.0632	1.5428	1.7592	37.5708	0.9965	2.7416	50.8316
		BA [18]	66.3891	1.7471	1.7295	37.3761	0.9966	2.6481	54.4752
		CSA [30]	63.4959	**0.7396**	1.7456	36.7040	0.9961	2.6844	51.4414
		SSA [29]	67.8002	1.3207	1.7384	38.3614	0.9967	2.3320	60.4296

Bold values refer to the best results

Table 9. Comparison the performance of the algorithms on the livingroom test image

Test Images	K	Algorithms	F_{avg}	Std	Time	PSNR	SSIM	MAE	SNR
Livingroom	2	EO	**12.6968**	**0.0000**	**0.1279**	**14.7322**	0.8072	**41.0498**	**1.9936**
		WOA [12]	**12.6968**	**0.0000**	0.1368	**14.7322**	0.8072	**41.0498**	**1.9936**
		HHA [28]	12.6966	0.0003	0.1331	14.7185	0.8075	**41.0498**	**1.9936**
		SCA [27]	12.6965	0.0004	0.1352	14.6965	0.8069	41.2280	1.9820
		PSO [18]	12.6968	0.0000	0.1399	14.7322	0.8072	**41.0498**	**1.9936**
		BA [18]	12.6964	0.0005	0.1347	14.7039	**0.8075**	41.3690	1.9728
		CSA [30]	12.6966	0.0003	0.1341	14.7072	0.8073	41.0942	1.9907
		SSA [29]	12.6968	0.0000	0.1347	14.7279	0.8072	**41.0498**	**1.9936**
	3	EO	**15.9376**	**0.0000**	**0.2683**	17.1624	0.8719	30.4610	2.8040
		WOA [12]	**15.9376**	**0.0000**	0.2756	17.1681	0.8719	30.3930	2.8127
		HHA [28]	15.9334	0.0027	0.2694	17.2310	0.8741	30.4571	2.8045
		SCA [27]	15.9348	0.0016	0.2761	**17.2492**	**0.8746**	**30.2079**	**2.8370**
		PSO [18]	15.9376	0.0000	0.2787	17.1624	0.8718	30.4610	2.8040
		BA [18]	15.9304	0.0065	0.2714	17.2960	0.8764	30.2226	2.8383
		CSA [30]	15.9342	0.0029	0.2719	17.2004	0.8726	30.2516	2.8317
		SSA [29]	15.9376	0.0000	0.2751	17.1624	0.8718	30.4610	2.8040
	4	EO	**18.9441**	**0.0007**	**0.4035**	19.2247	0.9249	24.0075	3.8222
		WOA [12]	18.9436	0.0011	0.4176	19.1255	0.9224	24.3719	3.7524
		HHA [28]	18.9280	0.0116	0.4066	19.2035	0.9235	24.2379	3.7769
		SCA [27]	18.9319	0.0077	0.4155	19.0132	0.9199	24.2548	3.7741
		PSO [18]	18.9440	0.0009	0.4202	19.2919	0.9261	23.7342	3.8791
		BA [18]	18.9163	0.0179	0.4160	**19.5176**	**0.9294**	**23.2231**	**3.9896**
		CSA [30]	18.9323	0.0067	0.4191	19.0657	0.9206	23.6329	3.9024

	SSA [29]	18.9419	0.0021	0.4134	19.3017	0.9257	23.5438	3.9175
5	EO	**21.7281**	0.0014	**0.5424**	**21.0818**	**0.9511**	**19.2964**	**4.9697**
	WOA [12]	**21.7281**	**0.0002**	0.5590	20.9807	0.9503	19.5994	4.8780
	HHA [28]	21.6701	0.0303	0.5465	20.9169	0.9474	19.5415	4.8854
	SCA [27]	21.7000	0.0171	0.5580	20.9027	0.9484	19.6433	4.8499
	PSO [18]	21.7265	0.0019	0.5611	21.0868	0.9512	19.7137	4.8225
	BA [18]	21.6723	0.0471	0.5574	20.9617	0.9479	20.0313	4.7344
	CSA [30]	21.6996	0.0201	0.5595	20.9802	0.9489	19.5840	4.8500
	SSA [29]	21.7128	0.0297	0.5626	20.9327	0.9491	19.6935	4.8431
10	EO	**33.8699**	0.0154	**0.6906**	25.8942	0.9818	**11.0400**	**9.1396**
	WOA [12]	33.8571	**0.0124**	0.7098	**25.8988**	**0.9819**	11.2482	8.9529
	HHA [28]	33.4419	0.1367	0.6942	25.2310	0.9775	11.5356	8.6921
	SCA [27]	33.4898	0.1050	0.7082	24.8109	0.9759	12.2510	8.0243
	PSO [18]	33.7946	0.0818	0.7135	25.9033	0.9818	11.9817	8.1788
	BA [18]	33.5225	0.1732	0.7088	25.6107	0.9800	11.1239	9.0412
	CSA [30]	33.4205	0.1413	0.7155	24.9912	0.9765	12.0274	8.2181
	SSA [29]	33.7467	0.1123	0.7124	25.8943	0.9817	11.0439	9.1194
15	EO	**43.6058**	**0.1002**	**0.8564**	**29.2047**	**0.9903**	7.5216	**13.5726**
	WOA [12]	43.4949	0.1130	0.8689	28.8816	0.9894	7.8488	12.8730
	HHA [28]	42.3966	0.2883	0.8705	27.6717	0.9855	8.0032	12.5645
	SCA [27]	42.4250	0.3735	0.8684	27.5469	0.9847	8.2607	11.9027
	PSO [18]	42.9211	0.4302	0.8767	28.9832	0.9890	8.7389	11.0836
	BA [18]	42.8895	0.3174	0.8752	28.6778	0.9885	7.8746	12.9079
	CSA [30]	42.4708	0.2420	0.8762	27.2774	0.9836	8.8033	11.1674
	SSA [29]	43.1396	0.2030	0.8788	29.1085	0.9897	**7.5129**	13.5396
20	EO	**51.5558**	**0.1832**	**1.0296**	**31.7127**	**0.9934**	**5.3899**	**18.8342**
	WOA [12]	51.3161	0.2371	1.0374	30.9499	0.9922	5.9213	17.1240
	HHA [28]	49.4925	0.3648	1.0384	29.5228	0.9892	6.0699	16.5132
	SCA [27]	49.4525	0.3941	1.0374	29.4804	0.9890	6.6024	14.5339
	PSO [18]	50.0175	0.5785	1.0561	30.7804	0.9913	6.8069	14.1161
	BA [18]	50.4389	0.5128	1.0452	30.7844	0.9917	6.0396	16.7055
	CSA [30]	49.5924	0.4530	1.0478	28.9705	0.9877	6.6110	14.5996
	SSA [29]	50.7400	0.2657	1.0494	31.5428	0.9929	5.4576	18.5665
30	EO	**63.6198**	**0.4144**	1.2331	**35.1143**	**0.9956**	**3.5340**	**27.1903**
	WOA [12]	63.5894	0.4337	1.2574	33.9376	0.9948	4.0362	23.8541
	HHA [28]	62.8503	0.6183	2.0899	33.8635	0.9946	4.3104	22.1448
	SCA [27]	59.6960	0.6308	1.2496	32.1560	0.9928	4.7556	19.5518
	PSO [18]	60.0761	0.9928	1.2688	33.2680	0.9938	4.6781	19.4601
	BA [18]	61.4662	1.1008	1.2542	33.3946	0.9939	4.0925	24.0502
	CSA [30]	59.5823	0.6793	**1.2282**	32.1964	0.9927	4.8298	19.3789
	SSA [29]	62.1823	1.0499	1.2470	34.0888	0.9947	3.7707	25.9677
40	EO	**71.9313**	1.6600	1.4570	**37.3507**	**0.9964**	**2.5113**	**36.4496**
	WOA [12]	71.2537	0.5200	1.4732	35.7206	0.9956	3.1616	29.0794
	HHA [28]	70.0848	0.8041	2.6660	35.0751	0.9951	3.1923	28.7817
	SCA [27]	66.1912	0.7321	1.4576	33.8732	0.9941	3.7389	24.4129
	PSO [18]	66.0772	1.0265	1.4940	35.2477	0.9952	3.5382	25.2955

	BA [18]	69.5373	1.3723	1.4680	35.4238	0.9952	3.2284	28.7147
	CSA [30]	66.3864	**0.6000**	**1.4326**	34.1921	0.9944	3.6406	25.4356
	SSA [29]	69.7134	1.2653	1.4513	35.4386	0.9953	2.8220	32.7381
50	EO	**77.3192**	1.6806	1.7108	**38.5571**	**0.9966**	**2.0025**	**42.5975**
	WOA [12]	76.2743	1.1282	1.7056	36.6438	0.9959	2.6686	34.0482
	HHA [28]	75.0848	0.9920	3.4180	36.4966	0.9958	2.5967	33.9291
	SCA [27]	70.8460	0.7490	1.6911	35.5688	0.9953	3.1374	28.4804
	PSO [18]	70.3306	0.7593	1.7410	36.6132	0.9958	2.8480	30.7864
	BA [18]	74.7284	1.5010	1.7035	36.7512	0.9959	2.7877	32.3518
	CSA [30]	71.0637	**0.7855**	**1.6526**	35.7084	0.9954	2.9257	30.0860
	SSA [29]	74.9288	1.3562	1.6754	36.8317	0.9960	2.3163	37.3122

Bold value refer to the best results

Table 10. Comparison the performance of the algorithms on the Peppers test image

Test Images	K	Algorith	F_{avg}	Std	Time	PSNR	SSIM	MAE	SNR
Peppers	2	EO	**12.5888**	**0.0000**	**0.1232**	16.6291	0.8829	**31.7885**	**2.6637**
		WOA [12]	12.5888	0.0000	0.1342	16.6291	0.8829	**31.7885**	**2.6637**
		HHA [28]	12.5887	0.0001	0.1295	16.6356	0.8831	**31.7885**	**2.6637**
		SCA [27]	12.5887	0.0001	0.1367	16.6364	0.8831	31.7974	2.6635
		PSO [18]	12.5888	0.0000	0.1305	16.6291	0.8829	**31.7885**	**2.6637**
		BA [18]	12.5878	0.0014	0.1305	**16.6541**	**0.8836**	31.8397	2.6612
		CSA [30]	12.5886	0.0002	0.1310	16.6357	0.8828	31.7913	2.6638
		SSA [29]	12.5888	0.0000	0.1373	16.6291	0.8829	**31.7885**	**2.6637**
	3	EO	**15.6215**	**0.0000**	**0.2517**	**18.6242**	**0.9325**	**26.1823**	**3.5389**
		WOA [12]	15.6215	0.0001	0.2689	18.5954	0.9321	26.2428	3.5312
		HHA [28]	15.6205	0.0005	0.2631	18.6052	0.9319	26.3038	3.5241
		SCA [27]	15.6208	0.0005	0.2709	18.5877	0.9318	26.3020	3.5219
		PSO [18]	15.6215	0.0000	0.2652	18.6017	0.9321	26.3524	3.5182
		BA [18]	15.6187	0.0038	0.2688	18.5430	0.9313	26.3501	3.5111
		CSA [30]	15.6207	0.0009	0.2693	18.5758	0.9318	26.3542	3.5127
		SSA [29]	15.6213	0.0001	0.2751	18.5644	0.9319	26.3999	3.5109
	4	EO	18.4494	0.0071	0**.3838**	19.9332	0.9482	21.9098	4.3210
		WOA [12]	**18.4508**	**0.0006**	0.4093	19.8941	0.9481	22.2245	4.2626
		HHA [28]	18.4385	0.0061	0.4020	19.7425	0.9468	22.1716	4.2777
		SCA [27]	18.4426	0.0069	0.4087	19.8245	0.9476	22.5824	4.1889
		PSO [18]	18.4478	0.0098	0.4041	19.9758	0.9485	22.3025	4.2465
		BA [18]	18.4383	0.0145	0.4061	19.9015	0.9479	**21.8491**	**4.3367**
		CSA [30]	18.4432	0.0046	0.4066	19.8484	0.9479	22.7789	4.1528
		SSA [29]	18.4449	0.0116	0.4108	**20.0304**	**0.9492**	22.1177	4.2819
	5	EO	**21.1693**	**0.0001**	**0.5200**	**21.7106**	**0.9641**	18.0654	5.3380
		WOA [12]	21.1690	0.0005	0.5476	21.7066	0.9639	18.1018	5.3256
		HHA [28]	21.1163	0.0230	0.5398	21.7039	0.9637	18.1032	5.3226
		SCA [27]	21.1380	0.0200	0.5470	21.6714	0.9639	18.1083	5.3374
		PSO [18]	21.1693	0.0001	0.5419	21.7021	0.9641	**18.0015**	**5.3550**
		BA [18]	21.1128	0.0394	0.5585	21.6956	0.9640	17.8181	5.4694
		CSA [30]	21.1421	0.0143	0.5444	21.6454	0.9639	17.9218	5.4166

	SSA [29]	21.1637	0.0040	0.5481	21.5962	0.9635	18.2227	5.2873
10	EO	32.4063	0.0437	**0.6635**	**27.1874**	**0.9891**	**9.5340**	**10.8434**
	WOA [12]	**32.4201**	**0.0221**	0.6953	27.1785	**0.9891**	9.6061	10.7497
	HHA [28]	31.7660	0.1746	0.6874	25.8822	0.9842	9.6250	10.6858
	SCA [27]	31.9880	0.1169	0.6973	26.3400	0.9861	10.2991	9.6948
	PSO [18]	32.4042	0.0318	0.6932	27.2133	0.9890	10.3883	9.5757
	BA [18]	32.0420	0.1678	0.7093	26.6182	0.9870	10.2325	9.8732
	CSA [30]	32.0107	0.1567	0.6926	26.4787	0.9865	10.6558	9.3119
	SSA [29]	32.3245	0.0786	0.6932	26.9909	0.9884	9.7695	10.5185
15	EO	**41.2360**	**0.0938**	**0.8263**	**30.1740**	**0.9934**	6.6596	15.6190
	WOA [12]	41.1888	0.0857	0.8559	29.7842	0.9928	6.8263	15.1860
	HHA [28]	39.8494	0.2445	0.8424	28.4265	0.9900	6.9158	14.8109
	SCA [27]	40.1784	0.2903	0.8559	28.8314	0.9908	7.2946	13.6714
	PSO [18]	41.0936	0.1548	0.8580	30.0598	0.9931	7.7510	12.6677
	BA [18]	40.3131	0.5381	0.8668	29.6475	0.9921	7.1281	14.2099
	CSA [30]	40.2436	0.3579	0.8533	29.0783	0.9912	7.9819	12.3201
	SSA [29]	40.7725	0.2362	0.8492	30.0856	0.9931	**6.5924**	**15.7075**
20	EO	**48.5733**	**0.1157**	**0.9932**	**32.5654**	**0.9952**	**5.0736**	**20.1785**
	WOA [12]	48.3582	0.1803	1.0223	32.2133	0.9949	5.2635	19.5614
	HHA [28]	45.9843	0.3643	1.0072	30.0083	0.9921	5.4047	18.8381
	SCA [27]	46.4947	0.6014	1.0259	30.5460	0.9929	5.9910	16.2680
	PSO [18]	47.7139	0.5054	1.0317	31.8114	0.9944	6.1661	15.9607
	BA [18]	46.8885	0.8549	1.0348	31.5688	0.9942	5.7620	17.6047
	CSA [30]	46.3308	0.6520	1.0192	30.7545	0.9931	6.3667	15.1631
	SSA [29]	47.6883	0.4391	1.0312	32.1926	0.9948	4.9610	20.7998
30	EO	**59.6414**	**0.2326**	1.2043	**35.9799**	**0.9967**	**3.3525**	**30.1369**
	WOA [12]	59.1248	0.4792	1.2506	34.6898	0.9961	3.6578	27.2176
	HHA [28]	58.2227	0.6521	2.0701	34.1370	0.9957	3.8185	25.6080
	SCA [27]	55.1098	0.8021	1.2350	33.2845	0.9951	4.1956	22.6756
	PSO [18]	56.9219	1.1480	1.2584	34.0792	0.9955	4.3074	21.7447
	BA [18]	57.3060	1.4578	1.2199	34.4318	0.9958	3.7768	26.1441
	CSA [30]	55.1240	0.6128	**1.1690**	32.7753	0.9948	4.3975	21.6828
	SSA [29]	58.0760	0.5824	1.2054	34.8641	0.9960	3.4322	29.1768
40	EO	**67.1298**	0.6528	1.4305	**38.2201**	**0.9972**	**2.4587**	**40.3209**
	WOA [12]	65.7456	0.9237	1.4560	36.7657	0.9967	2.8558	33.8796
	HHA [28]	64.4569	1.0671	2.6447	36.0325	0.9964	2.9302	33.1274
	SCA [27]	61.1123	1.1026	1.4446	35.3202	0.9962	3.2903	28.7042
	PSO [18]	62.2085	1.0950	1.4784	35.7658	0.9962	3.4577	26.8804
	BA [18]	64.0379	1.4614	1.4300	36.5026	0.9966	3.0205	32.0997
	CSA [30]	60.7316	**0.6828**	**1.3655**	34.6986	0.9958	3.4176	27.4474
	SSA [29]	65.4988	0.8189	1.4087	37.1967	0.9969	2.5610	38.5418
50	EO	**71.5093**	1.5573	1.6848	**39.6152**	**0.9973**	**1.9643**	**49.2164**
	WOA [12]	69.8383	1.0376	1.6874	38.3491	0.9971	2.3889	39.6639
	HHA [28]	68.8863	1.2787	3.3951	38.0555	0.9970	2.4221	38.8820
	SCA [27]	64.9364	0.9198	1.6708	36.9784	0.9967	2.7204	34.3867

	PSO [18]	65.4690	1.2947	1.7207	37.1265	0.9967	2.8030	32.7490
	BA [18]	68.0441	1.5134	1.6588	37.7690	0.9970	2.5946	36.7268
	CSA [30]	64.4661	**0.8080**	**1.5798**	36.5873	0.9966	2.7702	33.1970
	SSA [29]	69.1572	1.1999	1.6313	38.4245	0.9971	2.2443	42.7760

Bold value refers to the best results

5.5 Graphical performance evaluation

Figures 4-10 show the average of PSNR, SSIM, CPU time, Std, fitness values, MAE, and SNR respectively obtained by each algorithm on each threshold level. It is obvious from these figures that the proposed algorithm can outperform all other algorithms in PSNR, SSIM, fitness values, MAE, and SNR. On one hand, increasing the number of threshold levels, the Std, and CPU time obtained by the proposed algorithm increase compared with some other algorithms. Figures 11-17 show the average CPU time, fitness value, PSNR, SSIM, and Std respectively obtained by each algorithm on all threshold levels. Based on these figures, our proposed algorithm outperforms all other algorithms in PSNR, SSIM, fitness values, MAE, and SNR respectively. On the other hand, increasing the number of threshold levels, the Std, and CPU time obtained by the proposed algorithm increases compared with some other algorithms.

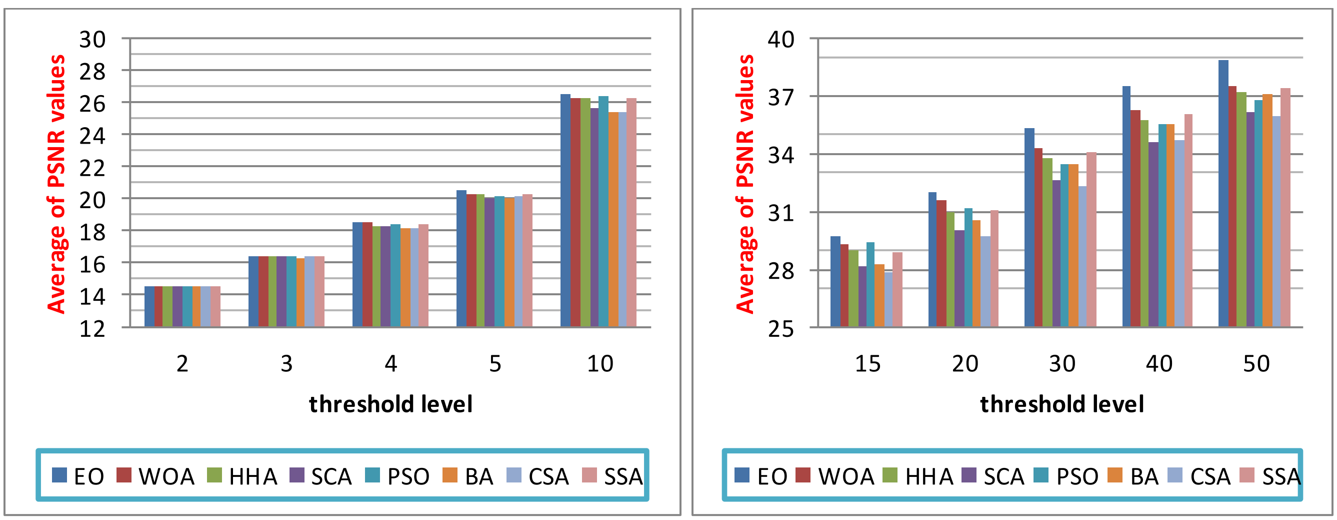


Fig.4 Average PSNR values on each thresholds level

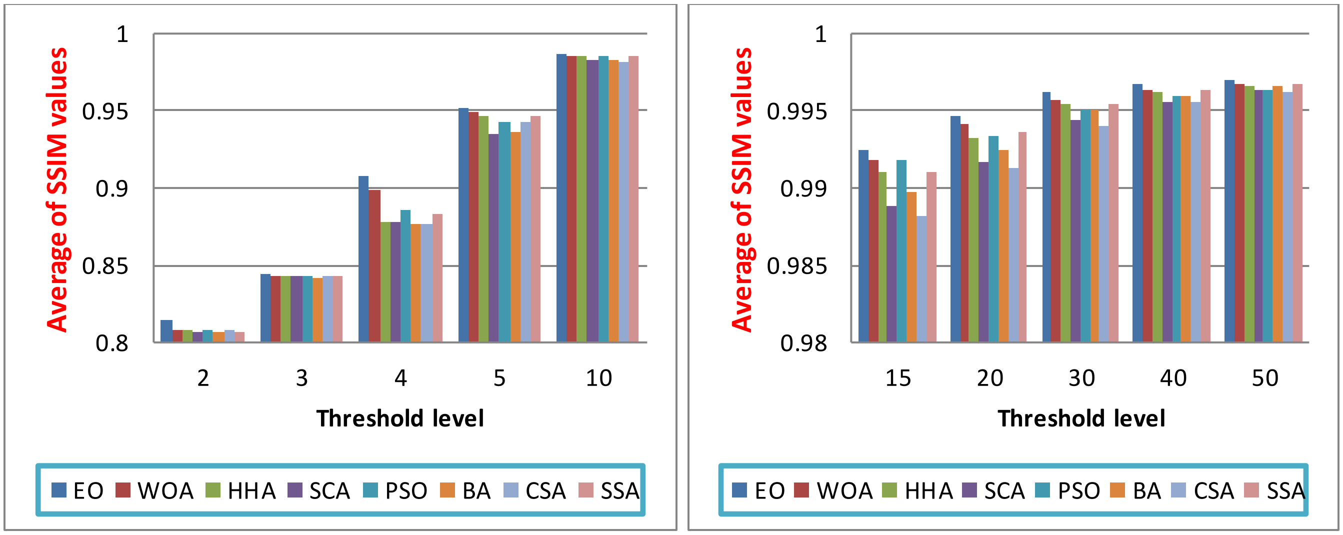


Fig.5 Average SSIM values on each thresholds level

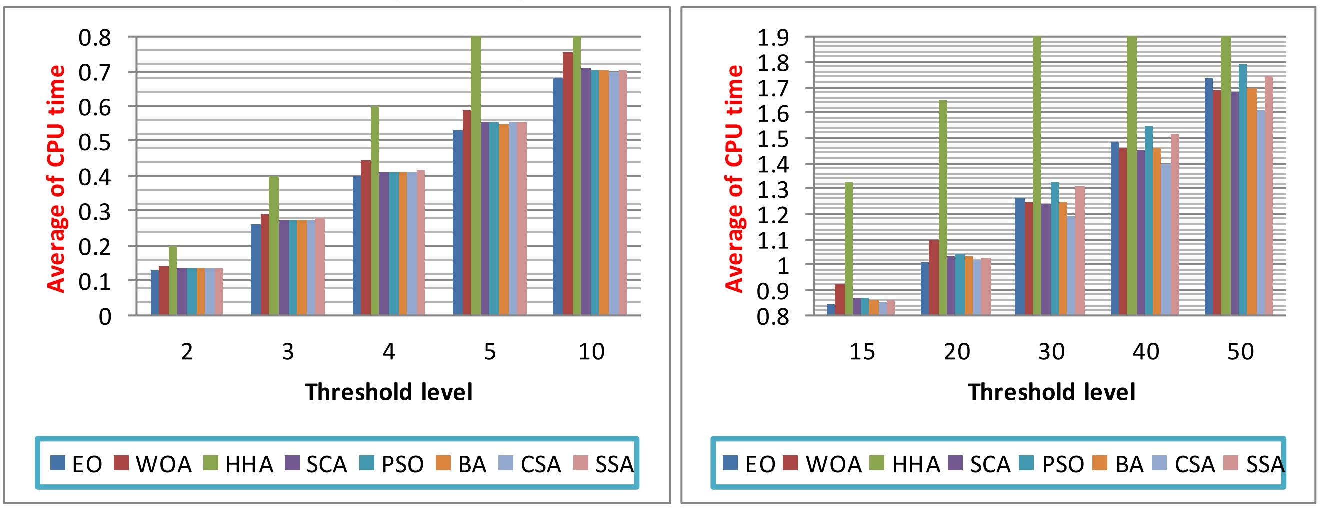


Fig.6 Average CPU Time values on each thresholds level

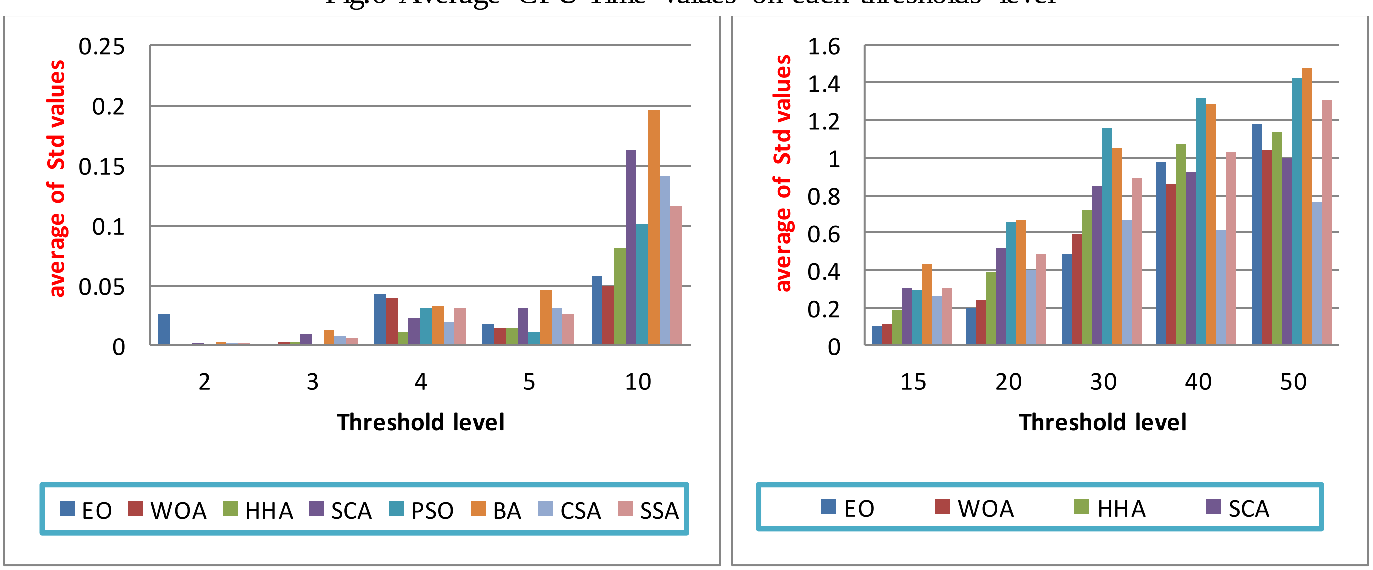


Fig.7 Average Std on each threshold level

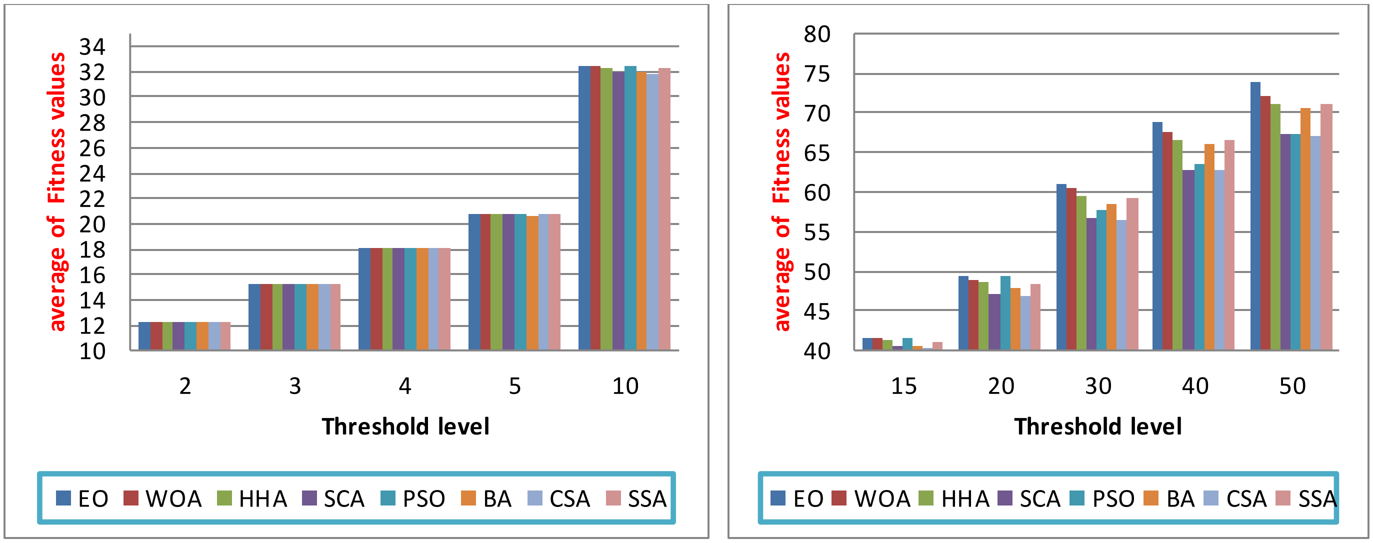


Fig.8 Average fitness values on each thresholds level

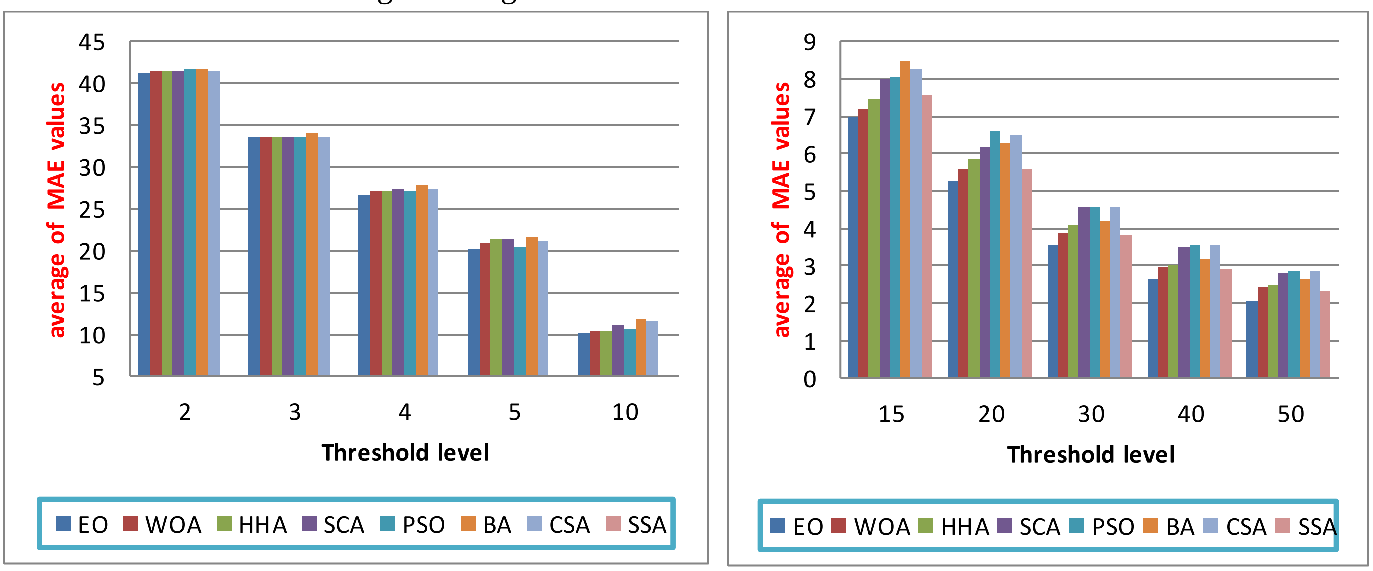


Fig.9 Average MAE values on each thresholds level

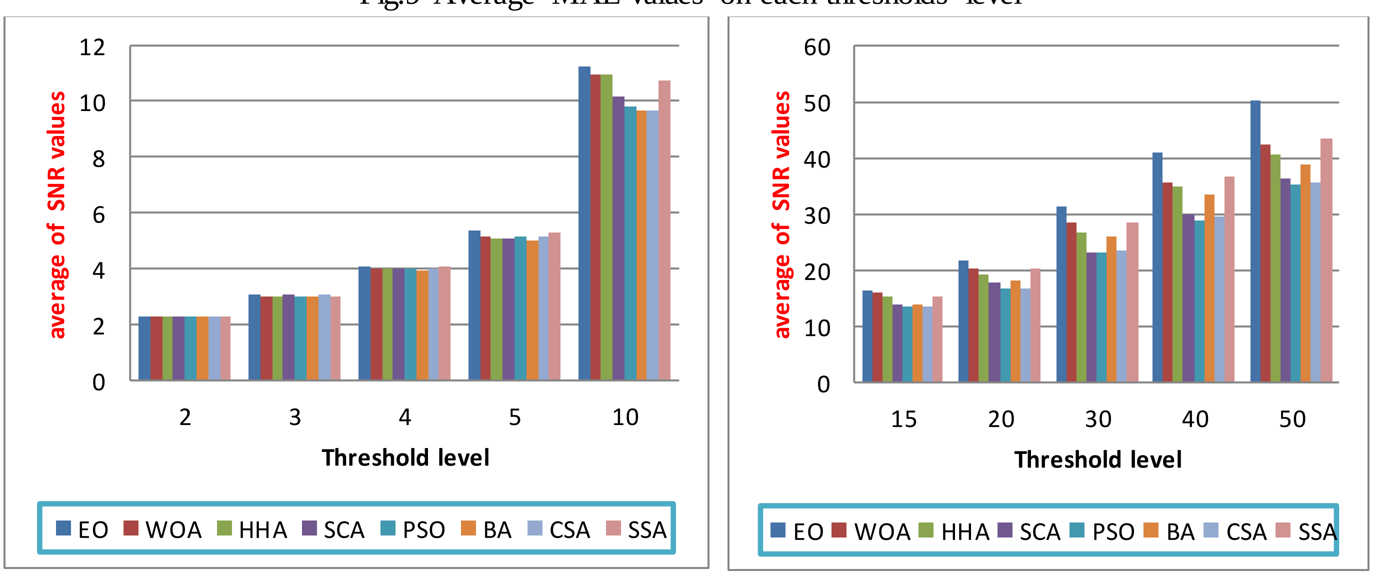


Fig.10 Average SNR values on each threshold level

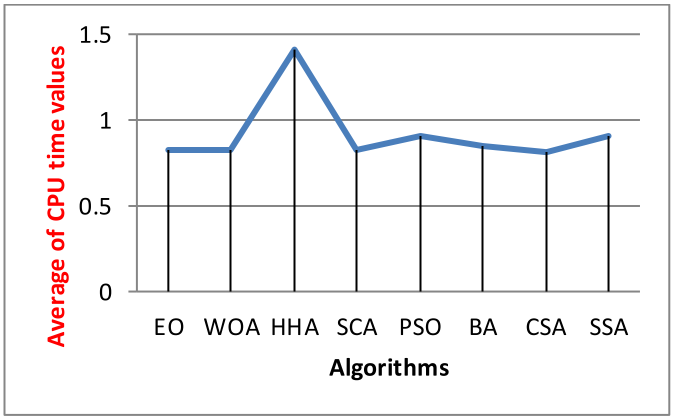


Fig.11 CPU time obtained by each algorithm

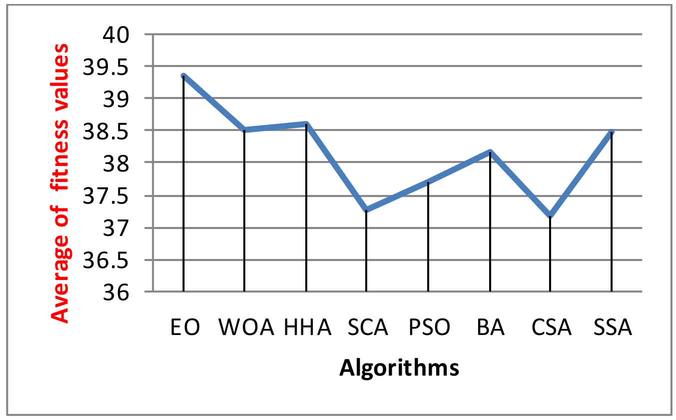


Fig.12 average fitness values obtained.

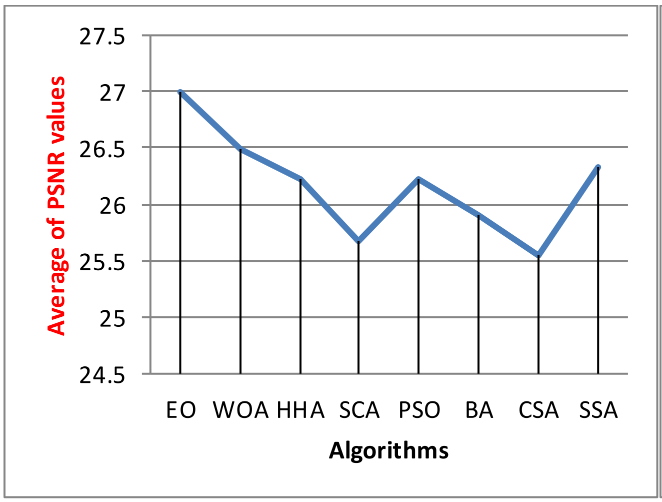


Fig.13 average of PSNR values obtained by each algorithm

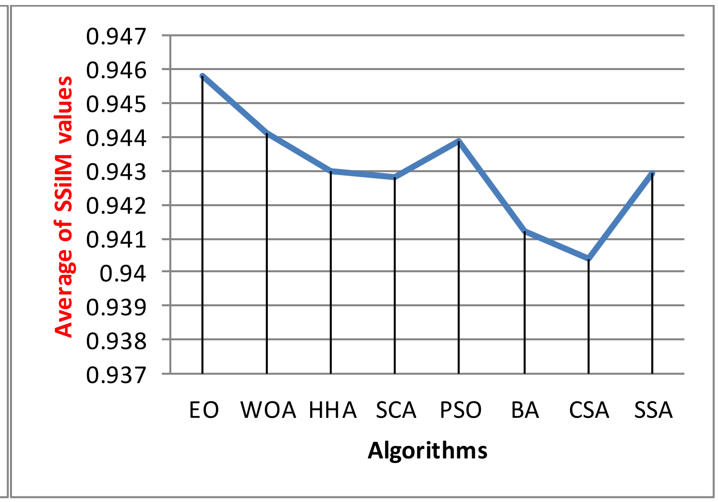


Fig.14 average of SSIM values obtained by each algorithm

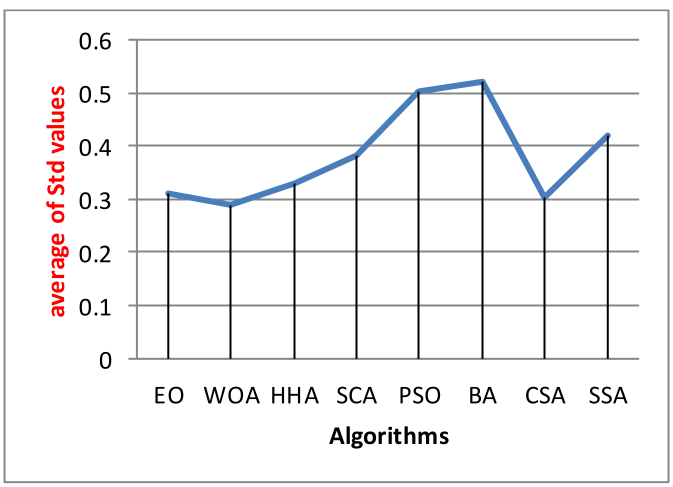


Fig.16 average Std values on all levels obtained by each algorithm

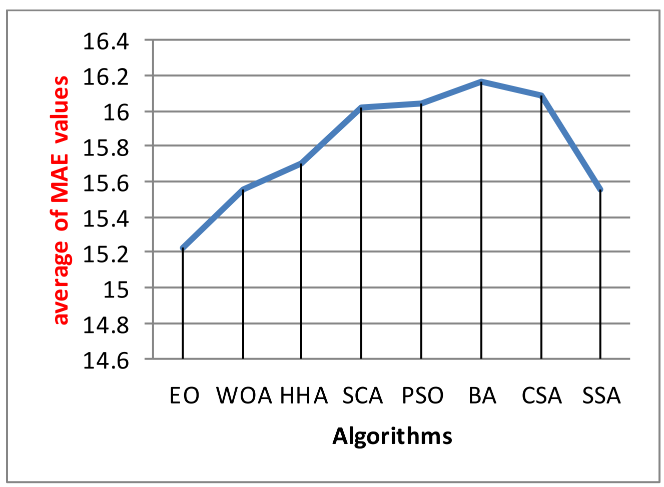


Fig.15 average MAE values on all levels obtained by each algorithm

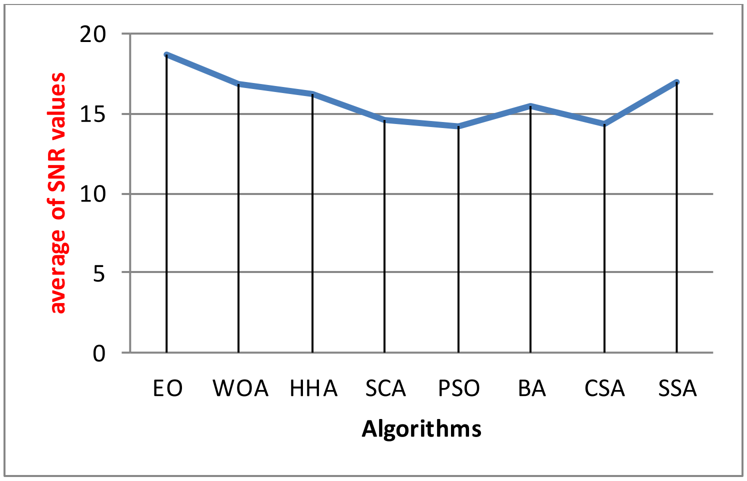


Fig.17 average SNR values on all levels obtained by each algorithm

5.6. **Mann Whitney U Test**

Wilcoxon rank-sum test\ Mann Whitney U Test [34] is a nonparametric test used to compare the results obtained by each pair of algorithms. This test is based on two hypotheses: the null hypothesis and the alternative hypothesis. The null hypothesis makes assumptions that there is no difference between the ranks of the results obtained by a pair of algorithms, and the alternative hypothesis considers that there is a difference between the ranks of the results obtained by a pair of algorithms. Wilcoxon rank-sum test is based here on a 5% significant level.

Table 11 shows the U and h values obtained by each pair of the algorithms. If U>0.05 or (h=0), then the null hypothesis is true, whereas if U<0.05 or (h=1), then the alternative hypothesis is true. Based on the result introduced in Table 11, our proposed algorithm can outperform all other algorithms.

Table 11. Results of Mann Whitney U Test for Kapur's method

		EO vs. WOA		**EO vs. SCA**		**EO vs. HHA**		**EO vs. CSA**		**EO vs. BA**		**EO vs. PSO**		**EO vs. SSA**	
Test images	**K**	**U**	**h**	**U**	**h**	**U**	**h**	**U**	**h**	**U**	**h**	**U**	**h**	**U**	**h**
CameraMan	2	>0.05	0	<0.05	1	>0.05	0	>0.05	0	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	4	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	10	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	15	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
house	2	>0.05	0	<0.05	1	>0.05	0	>0.05	0	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	4	>0.05	0	>0.05	0	>0.05	0	>0.05	0	>0.05	0	<0.05	1	>0.05	0
	5	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	<0.05	1	>0.05	0

	10	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	15	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
Lena	2	>0.05	0	>0.05	0	>0.05	0	>0.05	0	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	4	<0.05	1	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	10	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	15	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
Lake	2	>0.05	0	<0.05	1	>0.05	0	<0.05	1	>0.05	0	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	4	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	10	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	15	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	20	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
jetPlane	2	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	4	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	10	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	15	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
livingroom	2	>0.05	0	<0.05	1	>0.05	0	>0.05	0	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	4	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	10	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	15	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
	50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
peppers	2	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	3	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	>0.05	0
	4	>0.05	0	<0.05	1	>0.05	0	<0.05	1	<0.05	1	>0.05	0	<0.05	1
	5	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	>0.05	0	<0.05	1

10	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
15	>0.05	0	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
20	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
30	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
40	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1
50	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1	<0.05	1

6. Conclusion and Future work

Image segmentation could be considered the most important first step that should be performed accurately for image research analysis. Many techniques were proposed, such as Region-based, threshold-based, edge-based, and feature-based clustering, to resolve this research challenge. Due to the ease of use of the threshold-based segmentation, it was the preferred technique used for analyzing the image segmentation. To find the optimal threshold value for a grayscale image using the threshold technique, we used and improved on a novel meta-heuristic equilibrium algorithm due to its ability to addressing the massive scale problem with high efficiency. The performance of our algorithm was compared with seven existing algorithms like whale optimization algorithm (WOA), bat Algorithm (BA), sine-cosine algorithm (SCA), salp swarm algorithm (SSA), harris hawks algorithm (HHA), crow search algorithm (CSA), and particle swarm (PSO). The comparison was performed by applying the algorithms on a set of well-known test images taken from Berkeley Segmentation Dataset (BSD) with thresholds level between 2 and 50. Based on the results obtained by each algorithm, we concluded that the performance of our proposed algorithm was better than the other existing algorithms for the large thresholds level. Our algorithm, Equilibrium optimizer (EO), could outperform all other algorithms in the metrics used for evaluating the quality of segmented images for all thresholds level. However, EO could not outperform on some algorithms in Std values, and CPU time for the large thresholds level, as our main limitation.

Therefore, our future work includes improving the performance of EO by combining it with other meta-heuristic algorithms, the levy-flight strategy, or the opposition-based learning for reducing the running time and standard deviation values. Additionally, according to the high ability of EO for solving the large scale problem, we will test its performance on solving the combinatorial problems such as Multi-objective Knapsack problem, and flow shop scheduling.

Conflict of interest

The authors declare that there is no conflict of interest in the research.

Funding

This funding was supported by VC Research, VCR 0000016.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Kuruvilla, J., et al. *A review on image processing and image segmentation*. in *2016 international conference on data mining and advanced computing (SAPIENCE)*. 2016. IEEE.
2. Oliva, D., et al., *A multilevel thresholding algorithm using electromagnetism optimization.* Neurocomputing, 2014. **139**: p. 357-381.
3. Arora, S., et al., *Multilevel thresholding for image segmentation through a fast statistical recursive algorithm.* Pattern Recognition Letters, 2008. **29**(2): p. 119-125.
4. Hammouche, K., M. Diaf, and P. Siarry, *A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation.* Computer Vision and Image Understanding, 2008. **109**(2): p. 163-175.
5. Otsu, N., *A threshold selection method from gray-level histograms.* IEEE transactions on systems, man, and cybernetics, 1979. **9**(1): p. 62-66.
6. Kapur, J.N., P.K. Sahoo, and A.K. Wong, *A new method for gray-level picture thresholding using the entropy of the histogram.* Computer vision, graphics, and image processing, 1985. **29**(3): p. 273-285.
7. Abdel-Basset, M., et al., *A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem.* Future Generation Computer Systems, 2018. **85**: p. 129-145.
8. Sayed, G.I., A.E. Hassanien, and A.T. Azar, *Feature selection via a novel chaotic crow search algorithm.* Neural Computing and Applications, 2019. **31**(1): p. 171-188.
9. Rizk-Allah, R.M., et al., *A new binary salp swarm algorithm: development and application for optimization tasks.* Neural Computing and Applications, 2019. **31**(5): p. 1641-1663.
10. Elsayed, S.M., R.A. Sarker, and D.L. Essam, *A new genetic algorithm for solving optimization problems.* Engineering Applications of Artificial Intelligence, 2014. **27**: p. 57-69.
11. Guo, C. and H. Li. *Multilevel thresholding method for image segmentation based on an adaptive particle swarm optimization algorithm*. in *Australasian joint conference on artificial intelligence*. 2007. Springer.
12. El Aziz, M.A., A.A. Ewees, and A.E. Hassanien, *Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation.* Expert Systems with Applications, 2017. **83**: p. 242-256.
13. Kaveh, A. and S. Talatahari, *An improved ant colony optimization for constrained engineering design problems.* Engineering Computations, 2010. **27**(1): p. 155-182.
14. Erdmann, H., et al., *A study of a firefly meta-heuristics for multithreshold image segmentation*, in *Developments in medical image processing and computational vision*. 2015, Springer. p. 279-295.
15. Horng, M.-H., *Multilevel minimum cross entropy threshold selection based on the honey bee mating optimization.* Expert Systems with Applications, 2010. **37**(6): p. 4580-4592.
16. Agrawal, S., et al., *Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm.* Swarm and Evolutionary Computation, 2013. **11**: p. 16-30.
17. Ouadfel, S. and A. Taleb-Ahmed, *Social spiders optimization and flower pollination algorithm for multilevel image thresholding: a performance study.* Expert Systems with Applications, 2016. **55**: p. 566-584.
18. Chakraborty, F., D. Nandi, and P.K. Roy, *Oppositional symbiotic organisms search optimization for multilevel thresholding of color image.* Applied Soft Computing, 2019: p. 105577.
19. Chen, K., et al., *Multilevel image segmentation based on an improved firefly algorithm.* Mathematical Problems in Engineering, 2016. **2016**.
20. Bhandari, A.K., A. Kumar, and G.K. Singh, *Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur's, Otsu and Tsallis functions.* Expert Systems with Applications, 2015. **42**(3): p. 1573-1601.
21. Raja, N.S.M., S.A. Sukanya, and Y. Nikita, *Improved PSO based multi-level thresholding for cancer infected breast thermal images using Otsu.* Procedia Computer Science, 2015. **48**: p. 524-529.

22. Manikandan, S., et al., *Multilevel thresholding for segmentation of medical brain images using real coded genetic algorithm.* Measurement, 2014. **47**: p. 558-568.
23. Elaziz, M.A., A.A. Ewees, and D. Oliva, *Hyper-Heuristic Method for Multilevel Thresholding Image Segmentation.* Expert Systems with Applications, 2020: p. 113201.
24. Faramarzi, A., et al., *Equilibrium optimizer: A novel optimization algorithm.* Knowledge-Based Systems, 2019.
25. Mirjalili, S. and A. Lewis, *The whale optimization algorithm.* Advances in engineering software, 2016. **95**: p. 51-67.
26. Teodorović, D., *Bee colony optimization (BCO)*, in *Innovations in swarm intelligence*. 2009, Springer. p. 39-60.
27. Mirjalili, S., *SCA: a sine cosine algorithm for solving optimization problems.* Knowledge-Based Systems, 2016. **96**: p. 120-133.
28. Bao, X., H. Jia, and C. Lang, *A Novel Hybrid Harris Hawks Optimization for Color Image Multilevel Thresholding Segmentation.* IEEE Access, 2019. **7**: p. 76529-76546.
29. Wang, S., H. Jia, and X. Peng, *Modified salp swarm algorithm based multilevel thresholding for color image segmentation.* Mathematical biosciences and engineering: MBE, 2019. **17**(1): p. 700-724.
30. Askarzadeh, A., *A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm.* Computers & Structures, 2016. **169**: p. 1-12.
31. Hore, A. and D. Ziou. *Image quality metrics: PSNR vs. SSIM*. in *2010 20th International Conference on Pattern Recognition*. 2010. IEEE.
32. Rawat, V., *Investigation and assessment of disorder of ultrasound B-mode images.* arXiv preprint arXiv:1003.1827, 2010.
33. Chaurasia, K. and N. Sharma, *Performance Evaluation and Comparison of Different Noise, apply on TIF Image Format used in Deconvolution Wiener Filter (FFT) Algorithm.* IJCCER, 2014. **2**(4): p. 145-150.
34. Haynes, W., *Wilcoxon rank sum test.* Encyclopedia of systems biology, 2013: p. 2354-2355.
35. Pandey, H.M., *Performance evaluation of selection methods of genetic algorithm and network security concerns.* Physics Procedia, 2016. **78**: p. 13-18.
36. Pandey, H.M. *Performance Review of Harmony Search, Differential Evolution and Particle Swarm Optimization*. in *IOP Conference Series: Materials Science and Engineering*. 2017. IOP Publishing.
37. Pandey, H.M., A. Chaudhary, and D. Mehrotra, *Grammar induction using bit masking oriented genetic algorithm and comparative analysis.* Applied Soft Computing, 2016. **38**: p. 453-468.
38. Shukla, A., H.M. Pandey, and D. Mehrotra. *Comparative review of selection techniques in genetic algorithm*. in *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*. 2015. IEEE.
39. Pandey, H.M., A. Chaudhary, and D. Mehrotra, *A comparative review of approaches to prevent premature convergence in GA.* Applied Soft Computing, 2014. **24**: p. 1047-1077.