Multimedia Tools and Applications (2022) 81:33151–33170
https://doi.org/10.1007/s11042-022-13093-7

FOA: fireworks optimization algorithm

Ehsan Ehsaeyan[1] · Alireza Zolghadrasli[2]

Received: 26 January 2021 / Revised: 26 January 2022 / Accepted: 3 April 2022 /
Published online: 18 April 2022

Abstract
In this paper, a novel meta-heuristic algorithm called Fireworks Optimization Algorithm (FOA) is introduced with few control parameters for discrete and continuous optimization problems. This algorithm is inspired from explosion pyrotechnic devices producing colorful spikes like red, blue and silver. By modelling the explosion behavior of the Fireworks in the sky, the search space can be swept efficiently to find the global optima. To improve the balance between the exploration and exploitation of individuals, three categories are defined to avoid local optimal traps and applied to the search agents. Each category has a different task and predefined updating position rules. A grouping strategy is considered to prevent the algorithm from premature convergence. The performance of FOA is demonstrated over 15 standard benchmarks in the continuous version and 30 images thresholding problems in the discrete version. The obtained results reveal the superiority of the proposed algorithm with fewer input parameters over other state-of-the-art optimization methods in most cases.

Keywords Fireworks optimization algorithm · Bio-inspired algorithm · Benchmarked functions · meta-heuristic technique · Image segmentation · Engineering applications

1 Introduction

In real-life, there are many complicated problems which cannot be solved easily. Exact optimization algorithms may be chosen to solve these complex problems; However, high dimensionality and non-differentiability properties of hard-computing problems, make exact optimization algorithms unsuitable tools to achieve a good result. Hence, approximate

✉ Ehsan Ehsaeyan
ehsaeyan@sirjantech.ac.ir

Alireza Zolghadrasli
zolghadr@shirazu.ac.ir

[1] Electrical Engineering Department, Sirjan University of Technology, Sirjan, Iran

[2] Department of Communication and Electronic Engineering, Shiraz University, Shiraz, Iran

algorithms have been introduced to solve complex problems, faster and more reliable. Generally, searching algorithms are classified into two main categories: individual-based and population-based algorithms. There are some advantages and disadvantages for these two categories. For example, individual-based algorithms are fast in process and reliable for simple models. However, they depend on gradient information and cannot find global optimums in complex problems. Second group can fly from local optima better than first group and overcome complex environments with information sharing between individuals. However, they suffer from high computational cost compared to individual-based algorithms.

In literature, various meta-heuristic algorithms like physics-based [2, 19, 39, 46, 47], nature-based [6, 26, 34, 40, 48], animal-based [7, 22, 25, 42, 43], mathematics-based [31, 36, 41], evolutionary-based [13, 44] and virus-based [23, 32] algorithms have been reported. Physical laws of nature are basic ideas of physics-based algorithms which try to model physical relations to solve the problem. Nature-based algorithms are inspired by natural phenomena like lightning, ocean ecosystem, plant growth etc. The inspiration of animal-based algorithms is the behavior of animals in the co-operative foraging, mating fight etc. Mathematics-based algorithms are another group of meta-heuristic algorithms which consider the mathematic relations like gradient theorem, sine and cosine functions, … to find the solution. Evolutionary-based algorithms use natural selection or Darwinian theory to converge the global optimum. However, the implementation of Darwinian theory is different in this group of meta-heuristic algorithms. Virus-based algorithms are inspired by attacking, transmission and activity of virus in human bodies.

The mechanism of population-based algorithms is generally similar. The algorithm starts with initial candidate solutions. Usually, they are generated randomly with uniform distribution to cover the space. Then, the algorithm tries to find better locations in the searching process from generation to generation according to updating rules. Finally, the algorithm converges to the reasonable optimum by the end of run. However, there are some drawbacks to achieve the goal. Maintaining diversity is one of the major challenges of the meta-heuristic algorithms in solving complex problems. Also, premature convergence and becoming trapped in the local optima are other concerns of the optimization algorithms. To overcome these shortages, several algorithms have been developed in recent decades [24, 29].

Dhiman and Kaur introduced Sooty Tern Optimization Algorithm (STOA) to compromise between exploration and exploitation terms in the searching strategy [9]. To continue with the survey of articles, a new searching algorithm based on spiral movement of emperor penguins in the colony was proposed to solve different optimization problems [17]. Bouchekara developed a novel approach based on interaction between electric charged particles to design a circular antenna array [4]. Fathollahi-Fard et al. considered the mating behavior of the male red deer and suggested a new algorithm in this issue [12]. Black Widow Optimization (BWO) is another meta-heuristic approach which was reported for continuous nonlinear problems [21]. In literature, various algorithms are introduced in different issues by researchers. For example, Khare et al. proposed a hybrid classifier model for intrusion detection [30]. Recently, a novel optimizer inspired by barnacles mating behavior was introduced to solve reactive power dispatch problems [45].

In this paper, a novel meta-heuristic algorithm inspired by the explosion of fireworks in the sky, is introduced. We found out categorizing of the population helps the algorithm avoid trapping in local optima points and leads to faster convergence. Hence, three different categories are defined to cover most volunteer solutions of the global optimum and better sweep the search space. The proposed method is set by local optimum knowledge which has

been obtained before and tries to find better locations according to distance between sub-optimal points. A strict selection is considered to optimize the elitism strategy and vast random search is applied to reinforce the exploration. The innovations of our work are summarized as follows: 1) simplicity: FOA is as easy as well as robust searching algorithm. 2) efficient update rules: the location of light particles is updated with new rules which have not been reported before. 3) few control parameters: FOA has only two tuning parameters to explore searching space. Also, adapting formulas are suggested for these two parameters. 4) selection: new selection mechanism is employed in FOA algorithm.

The remainder of the paper is arranged as follows: The Fireworks Optimization Algorithm (FOA) is described in details and basic steps of the implementation are given in Section 2. The proposed algorithm is tested using famous continuous benchmarks and well-known test images and comparative results are presented in Section 3. Finally, Section 4 concludes this work.

2 Fireworks optimization algorithm

FOA is inspired by the explosion of pyrotechnic devices in the sky with production of noise, light and smoke. In some cases, explosion repeats several times from the beginning point like a branch (Fig. 1a). It means that a spike can blast and produce new spikes around itself and this exhibition continues to illuminate the sky. We model a population-based searching algorithm, named Fireworks Optimization Algorithm (FOA) from the behavior of fireworks explosion.

The step-wise implementation of FOA is described as follows:

Step 1: similar to other heuristic algorithms, initialization of the problem is the first step. Input range values (L_{min} and L_{max}), number of groups (G) and population size (N) are defined at this stage. The position of light particles is initialized as follows

$$\begin{aligned} &X\{i\} = L_{min} + (L_{max} - L_{min}) \times \boldsymbol{R} \\ &i = 1, \ldots, G \\ &\boldsymbol{R} = \left[r_{ij}\right]_{\frac{N}{G} \times K} \quad , \quad r_{ij} \in [0, 1] \end{aligned} \tag{1}$$

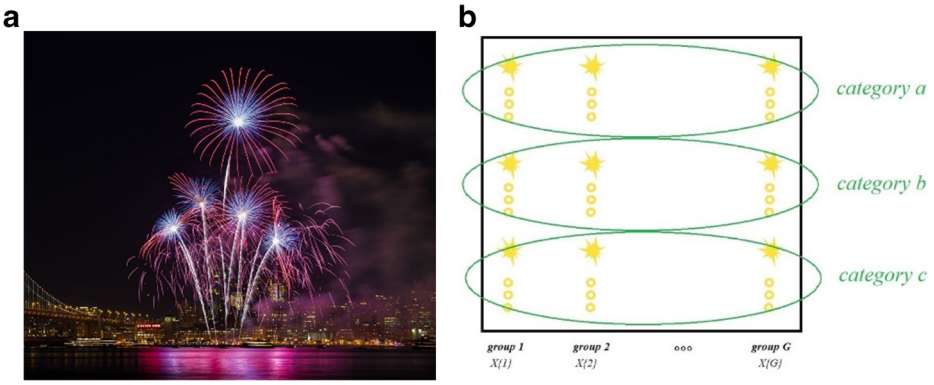


Fig. 1 **a** Fireworks display in the night sky, **b** population framework in FOA

Where $X\{i\}$ denotes the group number i which has $\frac{N}{G}$ members. $\boldsymbol{R}$ is random matrix with uniform distribution arrays which has $\frac{N}{G}$ rows and K columns. K is the dimension of the problem. The whole population $\boldsymbol{X}$ is aggregation of all groups.

$$\boldsymbol{X} = \bigcup_{i=1}^{G} X\{i\} = \begin{bmatrix} x_{11} & \cdots & x_{1K} \\ \vdots & \ddots & \vdots \\ x_{N1} & \ldots & x_{NK} \end{bmatrix} \tag{2}$$

Step 2: evaluate fitness of all individuals. The best position of a group named $Group_{Best}$ and the best position of all members is known as G_{Best}.

Step 3: every group has three categories: a, b and c as shown in Fig. 1b. Members belong to category a search the distance between G_{Best} and $Group_{Best}$. The position update mechanism of category a is different from discrete and continuous versions; because, in the discrete problem, the search space is countable and the exploitation is more reliable than exploration. Instead, in the continuous problem, more exploration is needed to reach a feasible solution. The second category is category b which tries to find valuable solutions between $Group_{Best}$s. Category c is the last category of the population which searches the area around G_{Best}. The category c is considered in the algorithm to maintain the diversity of the population.

In two categories a and b, positions are updated by $Group_{Best}$ and G_{Best}, which reduces the diversity of the whole population. The update rule of individuals is defined as

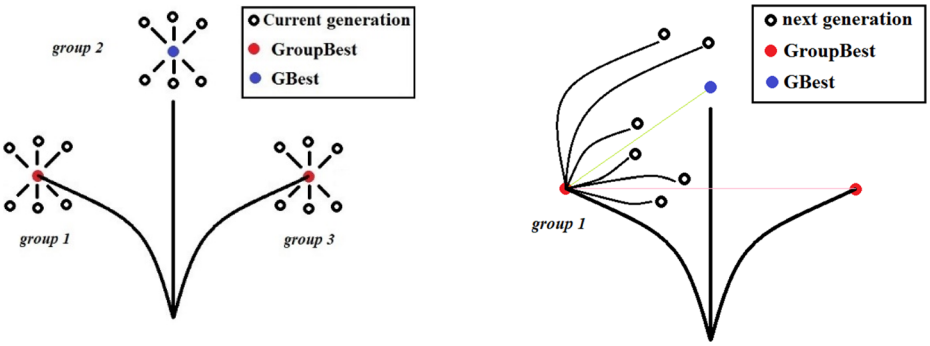
For discrete space

$$x_i(t+1) = \begin{cases} G_{Best}(t) + p_1 \times r_1 \times [G_{Best}(t) - Group_{Best\ i}(t)] & if\ x_i \in cat\ a \\ Group_{Best\ i}(t) + r_2 \times [Group_{Best\ m} - Group_{Best\ n}] & if\ x_i \in cat\ b \\ G_{Best}(t) + p_2 \times r_3 \times [L_{max} - L_{min}] & if\ x_i \in cat\ c \end{cases} \tag{3 - a}$$

For continuous space

$$x_i(t+1) = \begin{cases} p_1[r_1 \times (L_{max} - L_{min}) + r_2 \times (G_{Best}(t) - Group_{Best\ i}(t))] & if\ x_i \in cat\ a \\ Group_{Best\ i}(t) + r_3 \times [Group_{Best\ m} - Group_{Best\ n}] & if\ x_i \in cat\ b \\ r_4 \times G_{Best}(t) & if\ x_i \in cat\ c \end{cases} \tag{3 - b}$$

The convergence of the algorithm is guaranteed. When the algorithm finds a good solution, all light particles begin to approach G_{Best} and three so-called categories stop searching. To clarify the matter, we trace the population in two sequence iterations in Fig. 2. Three groups are considered in this figure. Every group has initial members and after the fitness calculation, G_{Best} and $Group_{Best}$s are assigned. Then, the only $Group_{Best}$s remains and generates new group population according to the distance between neighbor $Group_{Best}$ and G_{Best}. Figure 2b shows population generation of *group 1* in the next iteration.



(a) iteration t (b) iteration $t+1$ for group 1

Fig. 2 Population generation in two sequence iterations

According to above description, the pseudocode of FOA is shown in Fig. 3.

Similar to other natural-inspired algorithms, initialization is done by the random process. We used a grouping strategy to avoid premature convergence of the proposed algorithm. This technique was considered in various algorithms such as GGSA [10]. In FOA, the best individual is noted as G_{Best} and produces a new generation which acts as G_{Best} in PSO or *leader* in SSA [38]. However, update rules are different in concept. Elitism is employed in our algorithm to prune the offspring generation which some methods such as ADS [18] and ALO [35] have used this technique before. Also, in FOA control parameters (i.e., p_1 and p_2) have been adapted to the iteration number like WOA [37].

Start
Set parameters:
Maximum iteration *Maxiter*, population size *N*,
number of groups *G*, control parameters p_1,p_2
Initial generation:
Generate initial population randomly
Divide population to *G* groups
Calculate fitness of all members and specify G_{Best} and $Group_{Best}$
While iteration<Maxiter
For current group=1 **to** *G*
Generate light particle positions according to Eq 3
Correct infeasible solution
Calculate fitness of all individuals and reassign G_{Best} and $Group_{Best}$
Delete other members and retain G_{Best} and $Group_{Best}$
end
end
extract G_{Best}
Stop

Fig. 3 Pseudo code of FOA

3 Results and discussions

To evaluate the robustness of the introduced searching algorithm and compare it with other state-of-art natural-inspired methods, two experiments are done in this section. In the first experiment, image thresholding is considered. Multilevel thresholding is a segmentation method which partitions the image I into two or more subsets based on t threshold values. It is considered that a gray-scale image has L intensity levels. Multilevel thresholding can be defined as

$$\begin{aligned} R_0 &= \left\{ g(x,y) \in I | 0 \le g(x,y) \le t_1 - 1 \right. \\ &\vdots \\ R_K &= \left\{ g(x,y) \in I | t_K \le g(x,y) \le L - 1 \right. \end{aligned} \tag{4}$$

Where $g(x, y)$ is an image pixel, t_i $(i = 1, \ldots, k)$ indicates threshold value and K is the total threshold number. A criterion which is used in the multilevel thresholding is the energy function. To define the energy function of an image, a neighbourhood system of pixels must be introduced first. The neighbourhood mask N of order d for a pixel located at (i, j) is configured as $N_{pq}^{d} = \left\{ \left(i+u, j+v\right), (u,v) \in N^{d} \right) \}$ [15]. We only study the second order of neighbourhood (i.e $d = 2$). The second order neighbourhood mask N_{ij}^{2} is shown in Fig. 4.

Energy function is calculated for each gray level. For a given gray level l, a binary matrix B_l with the size of $m \times n$ (the same size as the original image) is built as $B_l = \{b_{ij},\ 1 \le \mathrm{i} \le m,\ 1 \le j \le n\}$ where $b_{ij} = 1$ if $g(x, y) > l$; else $b_{ij} = -1$. In other words, B_l indicates which pixel of the original image has lower or upper intensity than level l. Similarly, another matrix C is defined as $C = \{c_{ij},\ 1 \le \mathrm{i} \le m,\ 1 \le j \le n\}$ with elements of ones. i.e. $c_{ij} = 1,\ \forall\ (i, j)$. Energy function at gray level l is formulated as

$$E_l = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{pq \in N_{ij}^{2}} \left(c_{ij} c_{pq} - b_{ij} b_{pq} \right) \tag{5}$$

We define the matrix C to satisfy the positive energy condition $E_l \ge 0$. Kapur method is an entropy-based criterion which tries to make a centralized PDF distribution for each class on the segmented histogram [27]. The early algorithm is proposed for bi-level thresholding which tries to obtain an optimal threshold to extract the object from the background. Then this concept is applied to multilevel thresholding and utilized in many research studies. The thresholding problem can be mentioned as follows.

$$\text{Maximize } f(t_0, \ldots, t_K) = \textstyle\sum_{i=0}^{K} H_i$$

Fig. 4 Neighbourhood of central pixel (i, j)

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

Where

$$H_0 = -\sum_{i=0}^{t_1-1} \frac{p_i}{\omega_0} \ln \frac{p_i}{\omega_0}$$
$$\vdots$$
$$H_K = -\sum_{i=t_K}^{L-1} \frac{p_i}{\omega_K} \ln \frac{p_i}{\omega_K} \tag{6}$$

Eight famous methods, differential evolution (DE) [3], particle swarm optimization (PSO) [11], bat algorithm search (BAT) [5], flower pollination algorithm (FPA) [1], artificial bee colony (ABC) [28], harmony search (HS) [33], grey wolf optimizer (GWO) [20] and whale optimization algorithm (WOA) [14] with FOA are implemented to solve the image thresholding on the test images shown in Fig. 5 to prove the superiority of our proposed method. These methods are selected because of their good performance in previous works of image segmentation. These test images are well-known classical benchmarks used in the image processing literature. Figure 6 shows the related energy curve of images.

Kapur entropy is selected to compare the efficiency of different methods and the accuracy of the obtained solutions. To have an identical condition, the initial members of the searching methods are selected from the uniform distribution between [0, 255]. The goal is to find the best threshold values of four levels. To obtain a fair comparison between the searching algorithms, the number of iterations is set to 30, and the population size is assigned 150 for all methods. The control parameters of meta-heuristic algorithms are listed in Table 1. These parameters are chosen from original papers and best achieved by a trial-error procedure. The results are shown in Tables 2 and 3. The shown fitness is the average score of running each algorithm over 30 times and four threshold levels relate to the best optimal solution found by the algorithm over these runs. The peak signal to noise ratio (PSNR) is defined as

$$PSNR = 20\ log(255/RMSE) \tag{7 - a}$$

$$RMSE = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} (I(i,j) - I'(i,j))^2 / MN} \tag{7 - b}$$

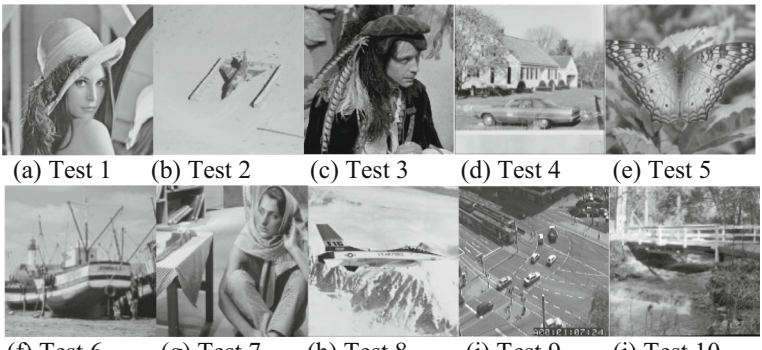


Fig. 5 Classical test images used to evaluate the proposed algorithm

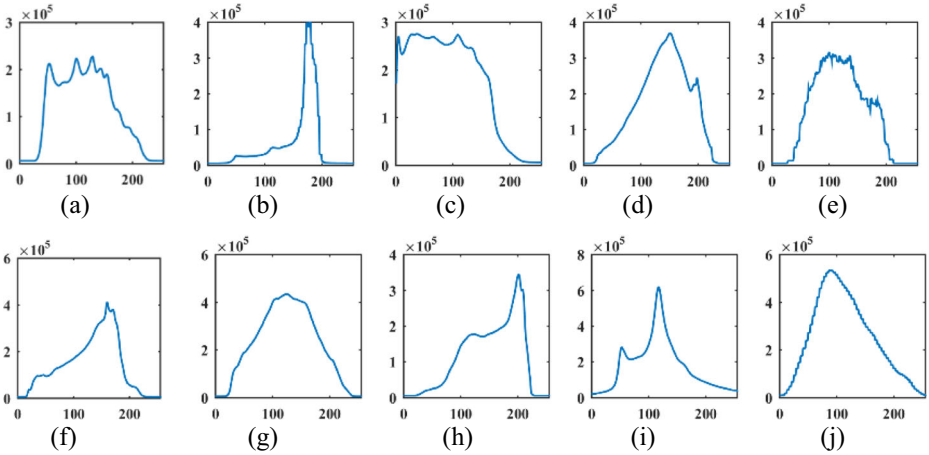


Fig. 6 Energy curve of test images. **a** energy curve of Test 1, **b** energy curve of Test 2, **c** energy curve of Test 3, **d** energy curve of Test 4, **e** energy curve of Test 5, **f** energy curve of Test 6, **g** energy curve of Test 7, **h** energy curve of Test 8, **i** energy curve of Test 9, **j** energy curve of Test 10

Where, N are the size of the test image, $I(i, j)$ and $I'(i, j)$ denote original and segmented images respectively and *RMSE* is the root mean-squared error between original and segmented images. SSIM describes the similarity of original and output images, which is defined as

$$SSIM = \frac{\left(2\mu_I\mu_{I'} + c_1\right)\left(2\sigma_{II'} + c_2\right)}{\left(\mu_I^2 + \mu_{I'}^2 + c_1\right)\left(\sigma_I^2 + \sigma_{I'}^2 + c_2\right)} \tag{8}$$

Where μ and σ are the mean and variance of I and I' images respectively. $\sigma_{II'}$ is the covariance between I and I'. c_1 and c_2 are the constants related to the pixel-values. Apparently, higher *PSNR* or *SSIM* indicates better image segmentation. These measurements increase with higher threshold values because the segmentation process is done resulting in higher precision.

Table 1 Control parameters for searching algorithms

Method	Parameters
ABC	limit=0.1×population
BAT	loudness=0.5
	pulse rate=0.9
	frequency range: [0,1]
DE	crossover probability=0.9
	scaling factor=0.8
FPA	probability switch=0.7
GWO	$\alpha = 2 - \frac{1}{max\ generation}$
HS	Consideration Rate=0.75
	Pitch adjusting rate=0.5
	new harmonies=0.1×population
PSO	velocity range: [−2,2]
	Cognitive constant=1.5
	Social constant=1.5
WOA	logarithmic spiral=1
FOA	$p_1 = \frac{current\ iteration}{2 \times max\ iteration}$
	$p_2 = \frac{1}{max\ iteration}$

Table 2 Obtained threshold results of different algorithms based on Kapur method (level = 4)

Image	Exhaustive search	ABC	BAT	WOA	FPA	GWO	HS	PSO	DE	FOA
test 1	33,98,162,219	33,103,164,220	33,110,163,214	33,98,162,219	31,101,163,220	33,98,162,219	34,92,152,222	33,97,161,219	33,97,165,218	33,98,162,219
test 2	43,101,155,199	38,100,158,199	41,99,153,199	44,99,153,199	43,98,152,201	43,99,155,199	39,105,160,200	43,101,155,199	42,103,155,200	43,101,155,199
test 3	58,115,172,213	58,122,175,215	58,115,172,213	57,116,172,213	52,117,170,210	57,115,172,213	56,103,172,208	58,115,172,213	57,116,173,213	58,115,172,213
test 4	60,113,169,224	59,111,168,225	49,102,159,224	59,109,167,224	52,112,171,224	59,112,167,224	61,113,169,226	60,113,168,224	62,113,167,224	60,113,169,224
test 5	38,94,149,205	37,97,149,205	37,94,149,205	38,96,149,205	38,88,153,212	38,93,149,205	38,102,153,206	38,94,149,205	37,94,149,206	38,95,151,212
test 6	69,127,186,221	69,130,184,219	65,123,180,216	74,128,186,220	66,127,192,222	67,126,186,221	63,122,183,221	69,127,186,221	75,133,188,222	69,127,186,221
test 7	23,78,133,187	22,83,141,192	70,121,172,220	23,79,136,188	23,72,127,178	23,77,132,187	24,70,128,186	23,77,132,186	22,80,140,191	23,77,132,187
test 8	64,116,168,223	68,118,177,223	62,112,163,223	65,116,172,223	58,101,165,223	64,115,168,223	60,118,164,223	64,117,169,223	61,109,161,223	64,116,168,223
test 9	42,96,149,202	39,93,145,201	41,96,152,203	41,99,149,203	39,97,142,191	41,96,148,201	44,102,155,201	42,96,149,202	40,95,152,206	42,96,149,202
test 10	53,104,153,202	49,99,147,197	53,104,153,202	53,106,155,202	52,97,142,193	53,103,152,202	59,101,149,201	53,103,153,202	49,100,151,201	53,104,153,202

Table 3 Obtained fitness results of different algorithms based on Kapur method (level = 4)

Image	Exhaustive search	ABC	BAT	WOA	FPA	GWO	HS	PSO	DE	FOA
test 1	19.21	19.17	18.89	19.18	19.11	**19.21**	19.09	19.05	19.17	**19.21**
test 2	19.4	19.31	19.28	19.35	19.28	19.39	19.15	**19.40**	19.37	**19.40**
test 3	19.42	19.38	19.41	19.40	19.35	**19.42**	19.34	**19.42**	19.41	**19.42**
test 4	19.04	18.99	18.77	18.93	18.96	19.03	19.00	18.83	19.02	**19.04**
test 5	19.28	19.26	19.02	19.26	19.21	**19.28**	19.23	19.15	19.26	**19.28**
test 6	19.03	18.95	18.86	18.95	18.92	**19.02**	18.95	18.99	18.97	**19.02**
test 7	18.79	18.74	18.70	**18.76**	18.73	18.74	18.73	18.74	18.74	**18.76**
test 8	19.17	19.13	19.05	19.07	19.11	19.16	19.13	19.02	19.14	**19.17**
test 9	19.43	19.40	19.40	19.37	19.34	**19.43**	19.35	**19.43**	19.42	**19.43**
test 10	19.1	19.08	19.03	19.08	19.05	**19.10**	19.07	**19.10**	19.08	**19.10**

Table 4 PSNR values using Kapur method for 4 levels thresholding

Image	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Exhaustive search	16.95	19.84	18.69	17.83	17.81	16.68	18.85	16.97	19.63	19.12
ABC	16.81	20.58	18.49	17.60	17.67	17.14	18.66	18.34	19.35	18.99
BAT	16.82	19.24	18.69	16.77	17.74	16.87	17.51	16.21	19.43	19.12
WOA	16.95	19.24	18.69	17.54	17.86	16.75	18.82	17.59	19.70	18.73
FPA	16.74	18.94	18.65	17.73	17.61	16.38	18.87	16.47	19.70	19.48
GWO	16.95	19.82	18.69	17.69	17.81	16.58	18.83	16.98	19.59	19.02
HS	17.53	21.07	18.37	17.72	17.39	16.38	18.63	16.25	19.66	19.09
PSO	16.99	19.84	18.69	17.79	17.81	16.68	18.86	17.11	19.63	19.06
DE	16.77	19.79	18.66	17.75	17.67	17.13	18.64	15.93	19.29	18.97
FOA	16.95	19.84	18.69	17.83	17.81	16.68	18.83	16.97	19.63	19.12

Tables 4 and 5 present PSNR and SSIM values of different algorithms respectively. We can observe from these tables that FOA gives better results than other algorithms; because, these algorithms converge in local sub-regions and cannot give satisfying threshold values. However, FOA achieves higher values in terms of objective function over 30 runs and outperforms other methods in exploring and exploiting the search space. Therefore, FOA is an efficient method for image multi-level thresholding as it does not waste time to search invaluable areas and manages the diversity of searching agents properly.

The segmented images based on Kapur method are shown in Figs. 7 and 8. The convergence curves of test images based on Kapur method are shown in Fig. 9 with 30 iterations.

Table 5 SSIM values using Kapur method for 4 levels thresholding

Image	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Exhaustive search	0.64	0.87	0.49	0.73	0.63	0.63	0.72	0.8	0.7	0.7
ABC	0.63	0.87	0.49	0.74	0.62	0.63	0.71	0.80	0.70	0.71
BAT	0.62	0.87	0.49	0.72	0.63	0.60	0.60	0.79	0.69	0.70
WOA	0.64	0.87	0.49	0.73	0.63	0.63	0.72	0.80	0.69	0.69
FPA	0.63	0.88	0.50	0.73	0.63	0.64	0.73	0.79	0.70	0.72
GWO	0.64	0.87	0.49	0.73	0.63	0.63	0.72	0.80	0.70	0.70
HS	0.66	0.87	0.50	0.75	0.62	0.61	0.72	0.79	0.67	0.70
PSO	0.64	0.87	0.49	0.73	0.63	0.63	0.72	0.80	0.70	0.70
DE	0.64	0.88	0.49	0.73	0.63	0.64	0.71	0.79	0.69	0.70
FOA	0.64	0.87	0.49	0.73	0.63	0.63	0.72	0.80	0.70	0.70

Fitness plots reveals that in most cases, FOA converges to the global optima in fewer iterations uniformly in comparison with other methods. Moreover, the proposed algorithm achieves higher Kapur fitness values in all test images. For example, for *test 1* the sequence is FPA < HS < WOA < PSO < ABC < BAT< DE < GWO < FOA according the increasing order of fitness.

Also, to prove the competency of the proposed algorithm, performance of FOA based multilevel thresholding method is tested by 20 images which have been selected from two MRI (Magnetic Resonance Imaging) and SAR (Synthetic-Aperture Radar) datasets and results

Fig. 7 Segmented images based on Otsu method (level = 4)

are compared with other methods. Two metrics, Kapur fitness and RMSE (Root Mean Square Error) are considered in this experiment. Figure 10 depicts these MRI and SAR test images.

For all the algorithms the population size is kept at 150 and number of iterations is set to 30 for K = 4. Table 6 reports mean fitness of Kapur entropy over 30 independent runs. The best results are marked in boldface. As seen in this table, FOA clearly outperforms its competitors except for *test 21* and *test 24*. RMSE measurements are computed through threshold value K = 4 and results are shown in Table 7. According to this table, FOA produces better results in 7 cases of 20 cases which achieves first rank while the second

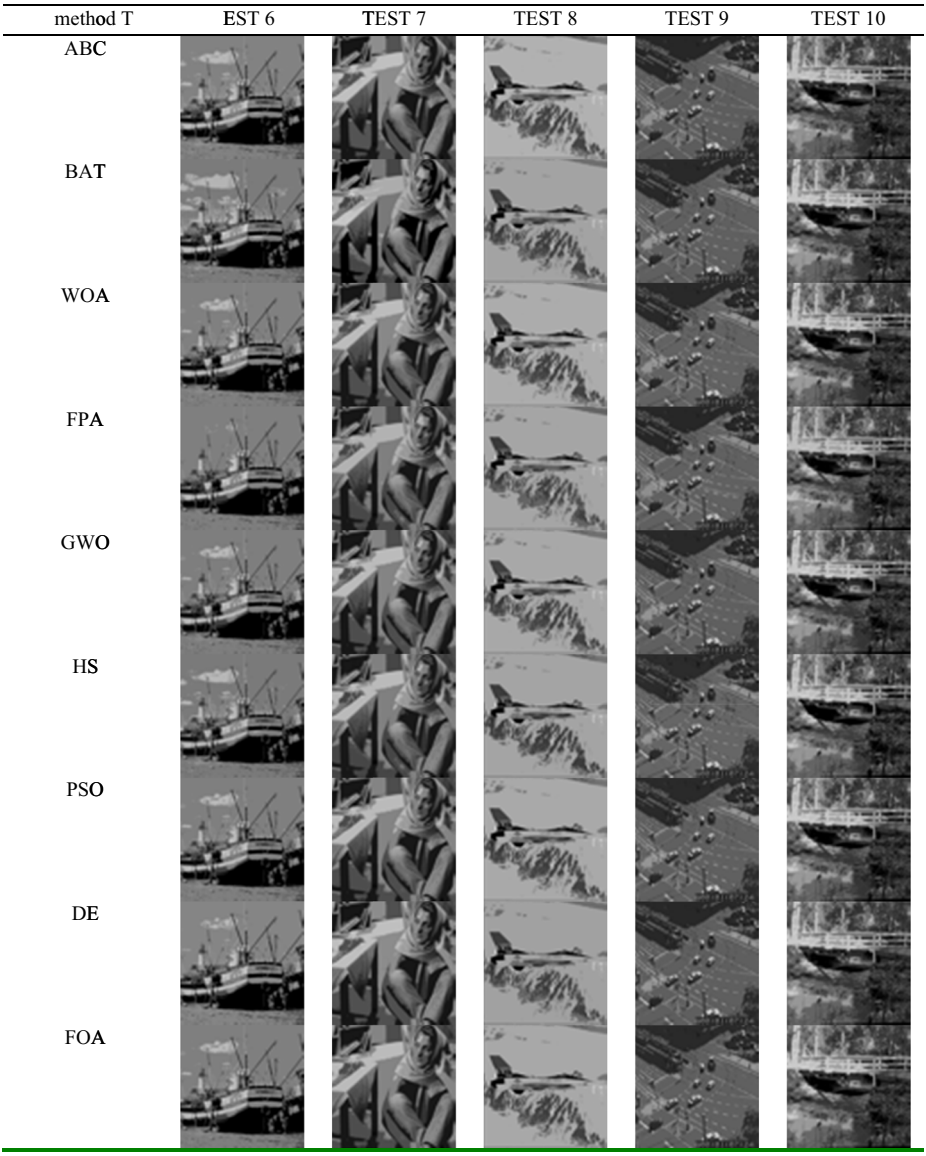


Fig. 8 Segmented images based on Otsu method (level = 4)

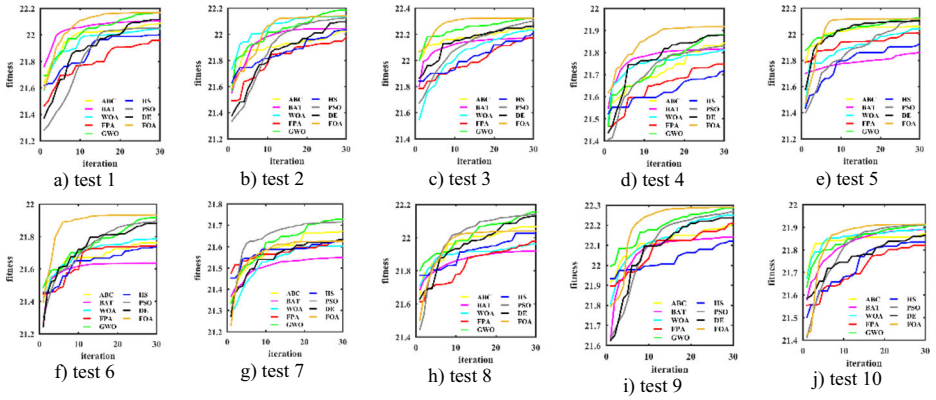


Fig. 9 Convergence curves based on Kapur method (level = 4)

rank goes to ABC. This means that FOA exhibits more detailed and accurate image information at threshold level 4.

In the next experiment, natural-inspired algorithms are examined by multimodal continuous problems and a number of optimization benchmarks are tested to evaluate the robustness of algorithms. Table 8 shows these benchmark functions with their global minimum. Similar to the first experiment, the population is set to 150 and 30 runs is considered. The results are shown in Table 9. According to this table, we concluded that FOA has an excellent performance in low dimensional problems and provides good solutions in most cases of high

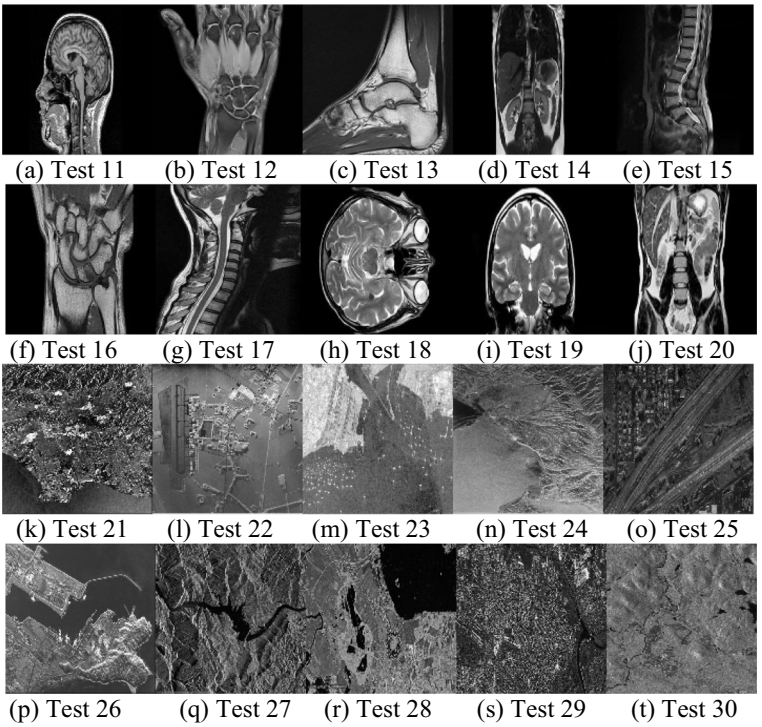


Fig. 10 Original test images, MRI images: test 11 to test 20, SAR images: test 21 to test 30

Table 6 Comparison of Kapur fitness values computed by various algorithms (level = 4)

Method	ABC	BAT	WOA	FPA	GWO	HS	PSO	DE	FOA
test 11	19.56	19.54	19.57	19.56	**19.59**	19.56	19.58	19.57	**19.59**
test 12	19.36	19.30	19.37	19.34	19.38	19.33	19.38	19.37	**19.39**
test 13	19.18	19.18	**19.21**	19.18	**19.21**	19.18	**19.21**	19.20	**19.21**
test 14	19.39	19.39	19.42	19.40	19.42	19.40	19.41	19.42	**19.43**
test 15	19.52	19.49	19.54	19.48	**19.55**	19.51	19.54	19.53	**19.55**
test 16	19.29	19.30	19.29	19.25	19.30	19.28	19.30	19.30	**19.31**
test 17	19.47	19.46	**19.49**	19.45	19.48	19.46	**19.49**	19.48	**19.49**
test 18	19.56	19.56	19.54	19.54	19.57	19.54	19.56	19.56	**19.57**
test 19	19.59	19.58	19.58	19.57	19.59	19.56	19.58	19.59	**19.60**
test 20	19.52	19.54	19.54	19.51	19.54	19.53	19.53	19.54	**19.55**
test 21	19.41	19.36	19.41	19.39	19.42	19.40	**19.43**	19.41	19.42
test 22	19.17	19.12	19.18	19.13	19.18	19.16	19.17	19.17	**19.19**
test 23	18.95	18.94	18.95	18.94	18.94	18.94	18.96	18.96	**18.97**
test 24	18.73	18.66	18.76	18.71	**18.77**	18.72	18.76	18.76	18.76
test 25	19.04	19.03	19.04	19.00	19.03	19.01	19.03	19.04	**19.05**
test 26	19.23	19.14	19.20	19.17	**19.25**	19.19	**19.25**	19.24	**19.25**
test 27	19.16	19.15	19.18	19.14	19.16	19.15	19.17	19.16	**19.18**
test 28	19.17	19.12	19.19	19.12	19.20	19.15	19.20	19.20	**19.21**
test 29	19.27	19.25	19.28	19.26	19.27	19.26	19.28	19.28	**19.29**
test 30	18.89	18.88	18.91	18.90	**18.92**	18.89	**18.92**	18.91	**18.92**

dimensional problems. It can be seen that FOA obtains the best results in 21 cases as first rank and WOA in 11 cases as the second rank. Other methods have not significant superiority over FOA and WOA. It means that FOA searches the space with a better sensitivity compared to other classical algorithms and converges to the global optimum faster than others; because, the mechanism of exploring is better organized and fewer individuals get stuck in the local optimums.

Table 7 Comparison of RMSE values computed by various algorithms (level = 4)

Method	ABC	BAT	WOA	FPA	GWO	HS	PSO	DE	FOA
test 11	17.84	18.15	18.15	18.11	18.30	18.57	18.15	18.07	**18.67**
test 12	23.50	22.82	24.08	23.89	23.56	23.34	23.67	**24.28**	23.67
test 13	27.00	27.40	**27.55**	27.48	27.42	27.10	27.40	27.44	27.40
test 14	19.08	19.38	19.66	19.52	19.67	19.89	19.89	20.39	**20.93**
test 15	20.39	20.60	20.69	**21.01**	20.61	20.71	20.60	20.72	20.60
test 16	21.92	21.88	22.31	21.93	21.92	22.38	22.11	21.99	**22.95**
test 17	**23.97**	23.55	23.53	23.43	23.61	23.42	23.51	23.62	23.59
test 18	**22.48**	21.77	21.88	22.08	21.89	21.28	21.89	22.06	21.89
test 19	20.02	20.13	19.96	20.44	20.16	20.29	20.40	20.01	**20.46**
test 20	21.32	21.23	21.30	21.29	21.23	**21.58**	21.23	21.19	21.23
test 21	29.38	29.18	29.35	29.33	29.32	29.18	29.18	28.14	**29.68**
test 22	31.84	31.84	31.54	31.20	31.47	30.24	31.84	31.64	**31.92**
test 23	29.88	29.94	30.12	**31.58**	29.95	30.29	29.95	31.39	29.95
test 24	**38.57**	36.66	36.85	36.27	36.34	34.57	36.66	36.05	36.66
test 25	33.96	35.70	35.14	36.93	35.69	36.16	35.70	33.34	**37.43**
test 26	24.67	24.82	24.83	26.10	25.05	**27.11**	25.24	24.20	25.24
test 27	29.21	28.38	**29.51**	26.92	29.31	28.70	29.31	28.87	29.31
test 28	26.59	26.80	25.91	**28.41**	26.18	25.67	26.39	26.22	26.39
test 29	**32.58**	30.55	30.22	30.50	30.58	32.15	30.58	31.61	30.58
test 30	29.03	28.92	28.55	28.45	28.92	**29.95**	28.81	29.42	28.81

Table 8 Benchmark functions with formula and global minimum for $\vec{x} = [x_1, \ldots, x_K]$

Function	Formula
BUKIN	$f\left(\vec{x}\right) = 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$
	f([−10, 1])=0
Cross in tray	$f\left(\vec{x}\right) = -0.0001\left(\left\lvert \sin(x_1)\sin(x_2)exp\left(\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert + 1\right)^{0.1}$
	f([±1.3491,±1.3491])=−2.06261
Drop wave	$f\left(\vec{x}\right) = -\frac{1+cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)} + 2$
	f([0,0])=−1
Bird	$f\left(\vec{x}\right) = \sin(x_1)e^{(1-\cos(x_2))^2} + \cos(x_2)e^{(1-\sin(x_2))^2} + (x_1 - x_2)^2$
	$f\left(\vec{x}\right) = -106.764537$
	x=[4.7,3.1], [−1.5,−3.1]
Schaffer N2	$f\left(\vec{x}\right) = 0.5 + sin^2(x^2 - y^2)\frac{-0.5}{[1+0.001(x^2+y^2)]}^2$
	f([0,0])=0
Akley	$f\left(\vec{x}\right) = -aexp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right) + a + \exp(1)$
	f([0 0…0])=0
GRIEWANK	$f\left(\vec{x}\right) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
	f([0 0…0])=0
Levy	$f\left(\vec{x}\right) = sin^2(\pi\omega_1) + \sum_{i=1}^{d-1}(\omega_i - 1)^2[1 + 10sin^2(\pi\omega_1 + 1)] + (\omega_d - 1)^2[1 + sin^2(2\pi\omega_d)]$
	$\omega_i = 1 + \frac{x_i - 1}{4} \qquad i = 1, \ldots, d$
	f([1 1…1])=0
RASTRIGIN	$f\left(\vec{x}\right) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10cos(2\pi x_i)\right]$
	f([0 0…0])=0
SCHWEFEL	$f\left(\vec{x}\right) = 418.9829d - \sum_{i=1}^{d} x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$
	f([420.9687 420.9687…])=0
Alpine N. 1	$f\left(\vec{x}\right) = \sum_{i=1}^{n}\lvert x_i sin(x_i) + 0.1x_i\rvert$
	f([0 0…0])=0
Happy Cat	$f\left(\vec{x}\right) = \left[\left(\lVert x\rVert^2 - n\right)^2\right]^{\alpha} + \frac{1}{n}\left(\frac{1}{2}\lVert x\rVert^2 + \sum_{i=1}^{n} x_i\right) + \frac{1}{2}$
	f([−1−1…−1])=0
Periodic	$f\left(\vec{x}\right) = 1 + \sum_{i=1}^{n} sin^2(x_i) - 0.1exp\left(\sum_{i=1}^{n} x_i^2\right)$
	f([0 0…0])=0.9
Salomon	$f\left(\vec{x}\right) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$
	f([0 0…0])=0
Styblinski-Tank	$f\left(\vec{x}\right) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4 - 16x_i^2 + 5x_i\right)$
	f([−2.903534−2.903534…])=−39.16599×d

A processing time comparison of various methods based on above benchmarks is considered to evaluate the convergence speed. Figure 11 shows the average running time of executing different algorithms over test functions. Based on obtained data, we conclude that

Table 9 Obtained results of executing different algorithms over 30 runs

Function	*d*	ABC	BAT	DE	FPA	GWO	HS	PSO	WOA	FOA
Bukin	2	0.13	0.27	1.46	0.93	0.38	0.17	5.43	0.72	**0.10**
cross in tray	2	−2.06	−2.03	−2.06	−2.06	−2.06	−2.00	−2.06	−2.06	**−2.06**
drop wave	2	−0.97	−0.90	−0.98	−0.96	−0.97	−1.00	−0.91	−0.98	**−1.00**
Bird Function	2	**−106.7**	−102.2	**−106.7**	**−106.7**	−106.0	−95.44	−97.37	−103.5	−106.0
Schaffer N2	2	0.06	0.15	0.03	0.12	0.00	0.00	0.47	0.00	**0.00**
Akley	20	12.21	11.54	19.69	16.41	0.12	0.00	20.14	0.00	**0.00**
	50	14.11	12.95	20.74	17.86	3.45	0.05	20.70	0.00	**0.00**
	100	15.47	13.23	20.98	18.00	6.17	4.17	20.88	0.00	**0.00**
Griewank	20	25.93	15.06	26.03	71.24	0.29	0.00	339.54	0.00	**0.00**
	50	99.45	55.98	411.69	265.35	1.55	0.00	1079.32	0.07	**0.00**
	100	237.3	140.2	1566.2	568.43	13.29	18.09	2388.5	0.00	**0.00**
Levy	20	7.62	58.28	20.04	41.02	1.01	11.61	57.88	1.81	**0.51**
	50	48.32	172.0	185.98	166.53	12.54	31.57	235.79	**1.00**	2.61
	100	135.7	411.1	540.88	364.17	44.79	68.49	551.26	**2.77**	6.31
Rastrigin	20	79.57	126.5	142.77	185.66	26.04	0.00	242.50	0.00	**0.00**
	50	322.9	453.9	586.14	544.38	140.9	0.00	720.13	7.54	**0.00**
	100	791.4	947.5	1403.23	113,330	433.1	22.30	1577.7	1.10	**0.00**
Schwefel	20	4930	5820	4568	5143	4399	6455	6315	**1067**	5018
	50	14,684	17,200	14,811	15,769	12,719	17,754	17,756	**2407**	15,443
	100	33,165	36,656	33,223	34,526	30,132	37,304	37,094	**4778**	32,474
Alpine N. 1	20	7.97	19.66	15.55	21.84	0.40	0.00	30.26	1.19	**0.00**
	50	36.38	71.15	78.16	67.30	7.32	0.00	99.71	0.02	**0.00**
	100	85.47	150.0	206.30	143.60	37.29	2.08	219.93	0.00	**0.00**
Happy Cat	20	0.76	1.23	0.62	0.81	0.61	2.23	2.25	**0.42**	0.73
	50	1.19	1.26	0.93	1.13	0.99	5.35	5.44	**0.53**	0.92
	100	1.34	1.27	1.24	1.30	1.13	8.16	8.42	**0.61**	1.02
Periodic	20	3.32	3.69	3.92	3.66	3.10	0.90	5.90	1.46	**0.90**
	50	13.11	13.75	12.87	11.21	8.90	0.90	16.72	1.93	**0.90**
	100	33.46	32.22	29.96	23.00	22.64	1.26	36.80	2.53	**0.90**
Salomon	20	6.32	5.01	6.77	9.37	0.60	**0.00**	18.73	0.14	0.07
	50	11.90	9.20	22.94	18.13	2.32	**0.02**	33.63	0.15	0.08
	100	17.21	13.98	43.61	26.44	6.24	4.18	49.38	0.20	**0.09**
Styblinski-Tank	20	−549.0	−460.9	−525.1	−470.2	−650.7	−279.4	−321.6	**−738.2**	−479.0
	50	−1119	−1027	−881	−939	−1285	−284	−440	**−1774**	−889.3
	100	−1932	−1857	−979	−1705	−2134	−298	−374	**−3690**	−1577

FOA has an accepting computation time in solving continuous problems. Taking into account of the computational time, we can line up them as BAT < PSO < HS < DE < WOA < GWO < FOA < FPA < ABC according to the increasing order of execution time. Although BAT ranks to be the first among them, it suffers the problem of converging to local optimum points.

Moreover, a statistical pair-wise test named Wilcoxon signed rank test is examined to determine the accuracy of the null hypothesis (same distribution of populations) [8]. Wilcoxon signed rank test returns *p-values* which should be less than 0.05 to reject the null hypothesis with 95% confidence. This test is done for various methods over 30 runs and results are registered in Table 10 for benchmark functions. The values more than 0.05 are boldfaced. From the overall comparison of obtained data, it can be stated that FOA performs significantly better than other meta-heuristic algorithms.

As seen in results, FOA is a powerful algorithm to find solutions in discrete and continuous problems. However, it has a drawback. To produce next generation, G_{Best} and $Group_{Best}$ remain and other individuals are deleted. Hence, a strict selection mechanism is applied to

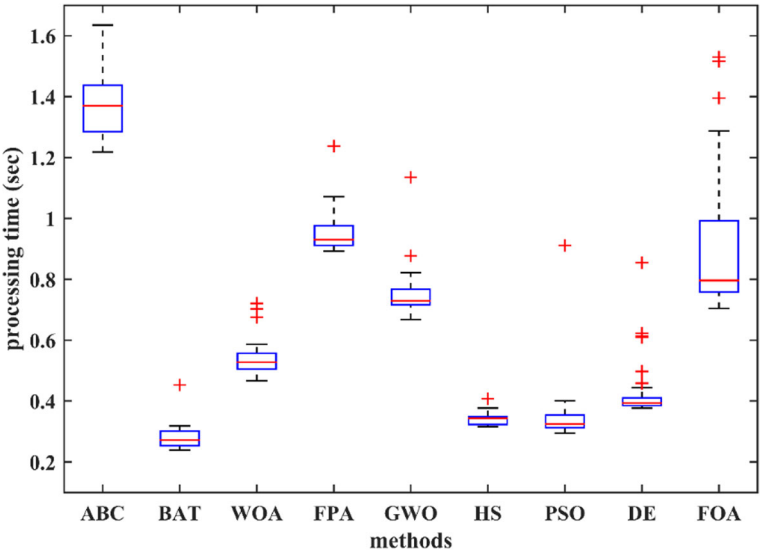


Fig. 11 Average running time of executing different algorithms

the population and valuable potential solutions maybe eliminated in high dimensional problems. In these cases, increasing population size or keeping some random individuals helps the population diversity. Also, changing update rules according to first and second rank of G_{Best} and $Group_{Best}$ may mitigate this limitation.

4 Conclusion

A simple and robust meta-heuristic algorithm inspired by the explosion of fireworks in the sky, was suggested in this paper. With taking advantage of grouping the population, fewer individuals were fallen in local optimums and premature convergence was mitigated. Another technique used in FOA is categorizing members. Three categories were considered which help the algorithm sweeping the search space very well and saving the time to explore invaluable areas. FOA was presented in two continuous and discrete versions which have similar mechanism of finding the global optimum and are different a little in the updating location of sparks.

The experiments are conducted on 30 images for subjective as well as objective assessments on 4-levels image thresholding segmentations in the discrete space. Also, the objective analysis is done on 15 state-of-the-art continuous benchmarks and compared with eight methods namely; ABC, BAT, WOA, FPA, GWO, HS, PSO and DE. The effectiveness of the new method has been studied in terms of convergence behavior which uses Kapur entropy as an objective function for extracting optimal thresholds. The experimental analysis revealed that the proposed method outperforms the existing compared methods for multi-levels image thresholding segmentation as well as objective functions in the continuous version. Compared to other algorithms, we can conclude that the computational complexity of FOA is better than WOA, ABC and GWO.

Three main research directions are foreseen for future work. First, the possibility of extending the proposed optimization algorithm to other image processing domains like image

Table 10 Comparison results with FOA *p-values* and other algorithms over test functions

Function	*d*	ABC	BAT	DE	FPA	GWO	HS	PSO	WOA
Bukin	2	6.0E-03	9.3E-06	1.7E-06	1.7E-06	1.7E-06	1.7E-06	1.7E-06	1.7E-06
Cross in tray	2	1.7E-06	1.5E-02	3.3E-04	1.8E-03	3.9E-04	1.7E-06	1.7E-06	**6.1E-01**
Drop wave	2	1.7E-06	1.7E-06	1.7E-06	1.7E-06	9.8E-04	1.7E-06	1.7E-06	1.8E-05
Bird Function	2	**1.9E-01**	6.6E-05	2.5E-05	8.2E-05	1.8E-03	1.7E-06	8.2E-05	**6.0E-02**
Schaffer N2	2	**8.2E-02**	1.7E-06	1.7E-06	1.7E-06	5.9E-05	1.7E-06	1.7E-06	3.8E-06
Akley	20	9.6E-05	8.7E-05	7.1E-05	9.0E-05	3.9E-05	8.0E-05	8.8E-05	9.9E-05
	50	3.2E-07	4.3E-05	4.7E-05	1.0E-05	7.3E-05	6.4E-05	1.5E-05	6.0E-05
	100	4.3E-05	5.8E-05	6.6E-05	6.4E-05	3.5E-05	2.9E-05	2.6E-05	1.1E-05
Griewank	20	8.2E-05	2.4E-05	5.8E-05	9.9E-05	9.2E-05	5.6E-05	7.8E-05	8.7E-05
	50	7.4E-05	7.7E-05	1.2E-05	4.4E-05	9.8E-06	7.3E-05	2.7E-05	6.8E-07
	100	5.8E-06	4.1E-05	5.1E-06	6.7E-05	9.6E-07	3.4E-05	1.0E-05	2.8E-05
Levy	20	1.7E-06	1.7E-06	**7.2E-02**	1.7E-06	2.2E-02	1.7E-06	1.7E-06	1.7E-06
	50	4.3E-05	4.3E-05	4.1E-05	9.5E-05	5.6E-05	9.7E-05	7.7E-05	1.9E-05
	100	3.1E-05	5.8E-05	8.9E-05	7.6E-05	8.6E-05	9.0E-05	2.7E-05	2.8E-06
Rastrigin	20	9.1E-05	8.9E-05	7.0E-05	1.7E-05	6.4E-05	1.6E-05	7.4E-05	9.1E-05
	50	5.0E-05	2.7E-05	7.3E-05	9.1E-05	5.0E-05	4.8E-05	5.9E-05	1.7E-05
	100	4.7E-05	4.5E-05	5.4E-05	8.5E-05	7.9E-05	1.2E-05	9.8E-05	5.3E-05
Schwefel	20	1.6E-02	5.8E-06	1.7E-06	**1.7E-01**	7.7E-06	1.7E-06	3.2E-06	4.7E-06
	50	1.6E-05	1.9E-06	1.7E-06	2.4E-02	6.9E-05	2.6E-06	1.7E-06	5.7E-03
	100	3.6E-05	3.9E-06	1.7E-06	1.7E-04	1.9E-04	1.7E-06	1.7E-06	**3.3E-01**
Alpine N. 1	20	6.0E-05	1.3E-05	1.4E-05	2.3E-05	8.9E-05	6.3E-05	7.1E-05	1.0E-05
	50	2.1E-05	4.9E-05	9.1E-05	5.1E-05	3.9E-05	1.6E-05	8.0E-05	3.0E-06
	100	1.7E-05	9.8E-06	8.5E-05	9.5E-05	2.0E-05	3.2E-05	1.4E-05	8.5E-05
Happy Cat	20	4.9E-02	1.4E-05	1.2E-05	7.8E-05	5.3E-05	1.7E-06	1.7E-06	**1.8E-01**
	50	2.0E-05	4.9E-05	5.2E-06	1.8E-05	9.3E-05	1.7E-06	1.7E-06	**5.7E-02**
	100	1.9E-04	8.9E-04	4.3E-06	7.9E-06	6.7E-06	1.7E-06	1.7E-06	1.2E-02
Periodic	20	8.2E-05	4.8E-05	3.5E-05	8.6E-05	7.8E-05	5.7E-05	7.5E-05	6.0E-07
	50	1.8E-05	8.0E-05	9.8E-05	9.4E-05	9.6E-05	2.6E-05	3.9E-06	5.5E-05
	100	5.9E-05	2.7E-05	4.0E-05	5.9E-06	6.6E-05	3.2E-05	5.3E-05	3.6E-05
Salomon	20	8.4E-05	3.1E-06	8.0E-05	2.3E-05	9.9E-05	7.8E-05	6.4E-05	6.5E-05
	50	9.8E-05	9.3E-05	3.7E-05	4.4E-05	7.1E-05	1.3E-05	1.2E-05	3.4E-05
	100	8.5E-05	8.2E-05	9.1E-05	2.3E-08	1.0E-05	4.8E-05	3.3E-05	5.6E-05
Styblinski-Tank	20	8.5E-06	3.3E-02	1.7E-06	**6.0E-02**	1.7E-06	1.7E-06	2.9E-06	1.4E-03
	50	1.7E-06	6.9E-05	1.7E-06	3.5E-02	1.7E-06	1.7E-06	1.7E-06	**7.8E-01**
	100	2.6E-05	4.6E-05	1.8E-05	2.6E-05	4.7E-05	2.9E-05	8.3E-05	8.9E-06

enhancement and classification can be worked out. Second, improving this algorithm in attaining its global optima using Minimum Cross entropy and Renyis entropy can be investigated for different image datasets. Another research direction is applying FOA to deep learning training and parameter optimization in the smoke detection application [16].

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abdel-Basset M, Shawky LA (2019) Flower pollination algorithm: a comprehensive review. Artif Intell Rev 52:2533–2557. https://doi.org/10.1007/s10462-018-9624-4
2. Anita YA (2019) AEFA: artificial electric field algorithm for global optimization. Swarm Evol Comput 48: 93–108. https://doi.org/10.1016/j.swevo.2019.03.013

3. Bilal PM, Zaheer H et al (2020) Differential evolution: a review of more than two decades of research. Eng Appl Artif Intell 90:103479. https://doi.org/10.1016/j.engappai.2020.103479
4. Bouchekara HREH (2020) Electric charged particles optimization and its application to the optimal design of a circular antenna array. Artif Intell Rev 54:1767–1802. https://doi.org/10.1007/s10462-020-09890-x
5. Chawla M, Duhan M (2015) Bat algorithm: a survey of the state-of-the-art. Appl Artif Intell 29:617–634. https://doi.org/10.1080/08839514.2015.1038434
6. Cheraghalipour A, Hajiaghaei-Keshteli M, Paydar MM (2018) Tree growth algorithm (TGA): a novel approach for solving optimization problems. Eng Appl Artif Intell 72:393–414. https://doi.org/10.1016/j.engappai.2018.04.021
7. de Vasconcelos Segundo EH, Mariani VC, dos Santos Coelho L (2019) Design of heat exchangers using falcon optimization algorithm. Appl Therm Eng 156:119–144. https://doi.org/10.1016/j.applthermaleng.2019.04.038
8. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1:3–18. https://doi.org/10.1016/j.swevo.2011.02.002
9. Dhiman G, Kaur A (2019) STOA: a bio-inspired based optimization algorithm for industrial engineering problems. Eng Appl Artif Intell 82:148–174. https://doi.org/10.1016/j.engappai.2019.03.021
10. Dowlatshahi MB, Nezamabadi-pour H (2014) GGSA: a grouping gravitational search algorithm for data clustering. Eng Appl Artif Intell 36:114–121. https://doi.org/10.1016/j.engappai.2014.07.016
11. Elbes M, Alzubi S, Kanan T, al-Fuqaha A, Hawashin B (2019) A survey on particle swarm optimization with emphasis on engineering and network applications. Evol Intel 12:113–129. https://doi.org/10.1007/s12065-019-00210-z
12. Fathollahi-Fard AM, Hajiaghaei-Keshteli M, Tavakkoli-Moghaddam R (2020) Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Comput 24:14637–14665. https://doi.org/10.1007/s00500-020-04812-z
13. Ghamisi P, Couceiro MS, Benediktsson JA, Ferreira NMF (2012) An efficient method for segmentation of images based on fractional calculus and natural selection. Expert Syst Appl 39:12407–12417. https://doi.org/10.1016/j.eswa.2012.04.078
14. Gharehchopogh FS, Gholizadeh H (2019) A comprehensive survey: whale optimization algorithm and its applications. Swarm Evol Comput 48:1–24. https://doi.org/10.1016/j.swevo.2019.03.004
15. Ghosh S, Bruzzone L, Patra S, Bovolo F, Ghosh A (2007) A context-sensitive technique for unsupervised change detection based on Hopfield-type neural networks. IEEE Trans Geosci Remote Sens 45:778–789. https://doi.org/10.1109/TGRS.2006.888861
16. Gu K, Xia Z, Qiao J, Lin W (2020) Deep dual-channel neural network for image-based smoke detection. IEEE Trans Multimedia 22:311–323. https://doi.org/10.1109/TMM.2019.2929009
17. Harifi S, Khalilian M, Mohammadzadeh J, Ebrahimnejad S (2019) Emperor penguins Colony: a new metaheuristic algorithm for optimization. Evol Intel 12:211–226. https://doi.org/10.1007/s12065-019-00212-x
18. Hasançebi O, Azad SK (2015) Adaptive dimensional search: a new metaheuristic algorithm for discrete truss sizing optimization. Comput Struct 154:1–16. https://doi.org/10.1016/j.compstruc.2015.03.014
19. Hashim FA, Houssein EH, Mabrouk MS, al-Atabany W, Mirjalili S (2019) Henry gas solubility optimization: a novel physics-based algorithm. Futur Gener Comput Syst 101:646–667. https://doi.org/10.1016/j.future.2019.07.015
20. Hatta NM, Zain AM, Sallehuddin R, Shayfull Z, Yusoff Y (2019) Recent studies on optimisation method of Grey wolf Optimiser (GWO): a review (2014–2017). Artif Intell Rev 52:2651–2683. https://doi.org/10.1007/s10462-018-9634-2
21. Hayyolalam V, Pourhaji Kazem AA (2020) Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng Appl Artif Intell 87:103249. https://doi.org/10.1016/j.engappai.2019.103249
22. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028
23. Jaderyan M, Khotanlou H (2016) Virulence optimization algorithm. Appl Soft Comput 43:596–618. https://doi.org/10.1016/j.asoc.2016.02.038
24. Jahani E, Chizari M (2018) Tackling global optimization problems with a novel algorithm – mouth brooding fish algorithm. Appl Soft Comput 62:987–1002. https://doi.org/10.1016/j.asoc.2017.09.035
25. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evol Comput 44:148–175. https://doi.org/10.1016/j.swevo.2018.02.013
26. Kaboli SHA, Selvaraj J, Rahim NA (2017) Rain-fall optimization algorithm: a population based algorithm for solving constrained optimization problems. J Comput Sci 19:31–42. https://doi.org/10.1016/j.jocs.2016.12.010

27. Kapur JN, Sahoo PK, Wong AKC (1985) A new method for gray-level picture thresholding using the entropy of the histogram. Comput Vis Graph Image Process 29:273–285. https://doi.org/10.1016/0734-189X(85)90125-2
28. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42:21–57. https://doi.org/10.1007/s10462-012-9328-0
29. Kaveh A, Kooshkebaghi M (2019) Artificial coronary circulation system; a new bio-inspired metaheuristic algorithm. Sci Iran. https://doi.org/10.24200/sci.2019.21366
30. Khare N, Devan P, Chowdhary C, Bhattacharya S, Singh G, Singh S, Yoon B (2020) SMO-DNN: spider monkey optimization and deep neural network hybrid classifier model for intrusion detection. Electronics 9: 692. https://doi.org/10.3390/electronics9040692
31. Kuo RJ, Zulvia FE (2015) The gradient evolution algorithm: a new metaheuristic. Inf Sci 316:246–265. https://doi.org/10.1016/j.ins.2015.04.031
32. Li MD, Zhao H, Weng XW, Han T (2016) A novel nature-inspired algorithm for optimization: virus colony search. Adv Eng Softw 92:65–88. https://doi.org/10.1016/j.advengsoft.2015.11.004
33. Manjarres D, Landa-Torres I, Gil-Lopez S, del Ser J, Bilbao MN, Salcedo-Sanz S, Geem ZW (2013) A survey on applications of the harmony search algorithm. Eng Appl Artif Intell 26:1818–1831. https://doi.org/10.1016/j.engappai.2013.05.008
34. Milan ST, Rajabion L, Ranjbar H, Navimipour NJ (2019) Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. Comput Oper Res 110:159–187. https://doi.org/10.1016/j.cor.2019.05.022
35. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
36. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96: 120–133. https://doi.org/10.1016/j.knosys.2015.12.022
37. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
38. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
39. Nematollahi AF, Rahiminejad A, Vahidi B (2017) A novel physical based meta-heuristic optimization method known as lightning attachment procedure optimization. Appl Soft Comput 59:596–621. https://doi.org/10.1016/j.asoc.2017.06.033
40. Nematollahi AF, Rahiminejad A, Vahidi B (2020) A novel meta-heuristic optimization method based on golden ratio in nature. Soft Comput 24:1117–1151. https://doi.org/10.1007/s00500-019-03949-w
41. Salimi H (2015) Stochastic fractal search: a powerful metaheuristic algorithm. Knowl-Based Syst 75:1–18. https://doi.org/10.1016/j.knosys.2014.07.025
42. Shadravan S, Naji HR, Bardsiri VK (2019) The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng Appl Artif Intell 80:20–34. https://doi.org/10.1016/j.engappai.2019.01.001
43. Shamsaldin AS, Rashid TA, Al-Rashid Agha RA et al (2019) Donkey and smuggler optimization algorithm: a collaborative working approach to path finding. J Comput Des Eng 6:562–583. https://doi.org/10.1016/j.jcde.2019.04.004
44. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359. https://doi.org/10.1023/A:1008202821328
45. Sulaiman MH, Mustaffa Z, Saari MM, Daniyal H (2020) Barnacles mating optimizer: a new bio-inspired algorithm for solving engineering optimization problems. Eng Appl Artif Intell 87:103330. https://doi.org/10.1016/j.engappai.2019.103330
46. Tahani M, Babayan N (2019) Flow regime algorithm (FRA): a physics-based meta-heuristics algorithm. Knowl Inf Syst 60:1001–1038. https://doi.org/10.1007/s10115-018-1253-3
47. Zhao W, Wang L, Zhang Z (2019) Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. Knowl-Based Syst 163:283–304. https://doi.org/10.1016/j.knosys.2018.08.030
48. Zhao W, Wang L, Zhang Z (2020) Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Comput & Applic 32:9383–9425. https://doi.org/10.1007/s00521-019-04452-x