Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5982410, 18 pages
https://doi.org/10.1155/2019/5982410

Research Article

A Novel Imperialist Competitive Algorithm for Multithreshold Image Segmentation

Mei Wang,[1] **Guohua Pan**,[2] **and Yan Liu**[1]

[1] *Laboratory of Image Processing & Pattern Recognition, Yantai Vocational College, Yantai 264670, China*
[2] *Yantai Public Security Bureau, Yantai 264670, China*

Correspondence should be addressed to Mei Wang; wangmei336@163.com

Received 30 March 2019; Revised 6 May 2019; Accepted 12 May 2019; Published 4 June 2019

Academic Editor: Federica Caselli

Multithreshold image segmentation plays a very important role in computer vision and pattern recognition. However, the computational complexity of multithreshold image segmentation increases exponentially with the increasing number of thresholds. Thus, in this paper, a novel imperialist competitive algorithm is proposed to solve the multithreshold image segmentation problem. Firstly, a new strategy of revolution and assimilation is adopted to improve the search efficiency of the algorithm. Secondly, imperialist self-learning and reserve country set are introduced to enhance the search of outstanding individuals in the population. Combining with the reserve country set, a novel imperialist competition strategy is proposed to remove the poorer individuals and improve the overall quality of the population. Finally, the sensitivity of the algorithm parameters is analyzed. Ten standard test pictures are selected to test. The experimental results show that the novel imperialist competitive algorithm has faster convergence speed, higher quality, and higher stability in solving multithreshold segmentation problems than methods from literature.

1. Introduction

Image segmentation is the technology of dividing an image into several specific or unique regions and extracting the interesting objects. It plays very important role in computer vision and pattern recognition. At present, thousands of image segmentation algorithms have been proposed [1–10], among which threshold segmentation is a better method and is used widely. Traditional threshold segmentation algorithm is very effective for single threshold segmentation, but with the increase of the thresholds number, the amount of computation will increase dramatically [3]. For a given image, the process of searching the optimal threshold can be regarded as a constrained optimization problem. In order to solve the problem of large amount of computation in multithreshold segmentation, many intelligent swarm optimization algorithms are used for image segmentation [11, 12], such as particle swarm optimization (PSO [13]), Gray Wolf algorithm (GWO [3]), teaching optimization algorithm (TLBO [14]), Cuckoo search algorithm [15], bee colony algorithm [16–18], and dragonfly based algorithm [19]. However, due to the limitations of the algorithm itself, the search ability is not ideal, and it is easy to premature convergence, fall into local optimum, and the solution stability is insufficient. [20]

ICA is a metaheuristic based on social and political behavior [21]. At present, the algorithm has been successfully applied to power grid energy management [22], nuclear reactor fuel loading [23], parameter estimation [24], production scheduling [25–29], and so on. According to literature [30], ICA not only is an effective global optimization method, but also has a strong neighborhood search capability. Its structure is flexible and easy to be fused with other algorithms. Thus, ICA is suitable for solving different problems with high stability. However, at present, ICA has not been applied to solve multithreshold segmentation problem. Therefore, a new imperialist competition algorithm in this paper has been given to improve the quality and stability of the multithreshold segmentation.

In this paper, a new imperialist competition algorithm is proposed to solve the multithreshold image segmentation problem. Firstly, a new strategy of revolution and assimilation is adopted to improve the search efficiency of the algorithm. Then, in order to enhance the search for excellent individuals in the population, imperialist self-learning and reserve country set are introduced. Thirdly, a new imperialist competition strategy combining with the reserve country set is proposed to weed out the poorer individuals and improve the overall quality of the population. Finally, the sensitivity of the algorithm parameters is analyzed. Ten standard test pictures are selected to test the performance comparing with three efficient metaheuristics, and the experimental results show that the novel imperialist competitive algorithm has faster convergence speed, higher quality, and higher stability in solving multithreshold segmentation problems.

The remainder of the paper is organized as follows. Problem under study is described in Section 2. ICA for the problem is reported in Section 3. Numerical test experiments on ICA are shown in Section 4 and the conclusions are summarized in the final section and some topics of the future research are provided.

2. Problem Description

Threshold segmentation method is used commonly for image segmentation and the traditional threshold selection method is described as Otsu [1] and Kapur's [2] method. Assume an image gray level is L and it is shown as $\{0, 1, 2, \ldots, (L-1)\}$. N is the pixels number of the image and n_i is the pixel number of gray value i. $p = \{p_i \mid i = 0, 1, 2, \ldots, (L-1)\}$ is described as the probability distribution of each gray value i, where $p_i = n_i/N$.

2.1. Otsu Method. The classical Otsu method [1] is known as the optimal threshold selection method. And Otsu for multiobjective image segmentation is the method to determine the optimal segmentation threshold group based on the maximum between-class variance criterion. The traditional Otsu's method is described as follows.

Assume the image gray level is L, n_i is the pixels number of gray-level i, and the total pixels number of the image is $N = \sum_{i=0}^{L-1} n_i$. If the probability of a given gray-level i is given as $p_i = n_i/N$, obviously, $p_i \geq 0$, $\sum_{i=0}^{L-1} p_i = 1$. Assume the image segmentation threshold is θ_t, g_0 shows the average gray-level value of all pixels less than or equal to θ_t, and g_1 shows the average gray-level value of all pixels larger than θ_t in the image. Then variance between binary image classes is described as

$$\sigma_B^2(t) = \omega_0\sigma_0^2 + \omega_1\sigma_1^2 = \omega_0\left(g_0 - \theta_t\right)^2 + \omega_1\left(g_1 - \theta_t\right)^2 \quad (1)$$

where $\omega_0 = \sum_{i=0}^{t-1} p_i$, $\omega_1 = \sum_{i=t}^{L-1} p_i = 1 - \omega_0$. $\sigma_B^2(t)$ can be defined as a reparability criterion. So θ_t can be computed by maximizing $\sigma_B^2(t)$.

The formula (1) can be extended to multithreshold segmentation (assuming M level). Threshold group $[t_1, t_2, \ldots, t_{M-1}]$ can divide one image into k-classes objectives. And the variance between classes can be described as follows:

$$\sigma_B^2\left(t_1, t_2, \ldots, t_{M-1}\right) = \sum_{j=0}^{M-1} \omega_j\left(g_j - \theta_j\right)^2 \quad (2)$$

The threshold group $[t_1, t_2, \ldots, t_{M-1}]$ of multiobjective image segmentation can be given by maximizing the between-classes variance $\sigma_B^2(t_1, t_2, \ldots, t_{M-1})$.

2.2. Kapur's Method. Optimum entropy threshold method presented by Kapur [2] can be described as follows. For gray images, the grayscale also can be expressed as $\{0, 1, 2, \ldots, (L-1)\}$ $L = 256$. n_i shows the pixels number of the image with gray-value i. N shows the total pixels number of the image and $N = \sum_{i=0}^{L-1} n_i$. The probability density is $p_i = n_i/N$ and the entropy of the image is defined as $H_n = -\sum_{i=0}^{L-1} p_i \ln p_i$.

For image multilevel segmentation (assuming M level), the image can be divided into M subclasses: $C_0 = \{0, 1, \ldots, t_1\}$, $C_1 = \{t_1 + 1, t_1 + 2, \ldots, t_2\}, \ldots, C_{M-1} = \{t_{M-1} + 1, t_{M-1} + 2, \ldots, L-1\}$, and the entropy is defined as

$$\begin{aligned}\varphi\left(t_1, t_2, \ldots, t_{M-1}\right) = &-\sum_{i=0}^{t_1} \frac{p_i}{P_0} \ln \frac{p_i}{P_0} - \sum_{i=t_{1+1}}^{t_2} \frac{p_i}{P_1} \ln \frac{p_i}{P_1} \\ &- \cdots - \sum_{i=t_{M-1}}^{L-1} \frac{p_i}{P_{M-1}} \ln \frac{p_i}{P_{M-1}}\end{aligned} \quad (3)$$

where $P_j = \sum_{i\in C_j} p_i$, $j = 0, 1, \ldots M-1$.

The optimum entropy threshold selection method proposed by Kapur does not need prior knowledge. It can be used to segment nonideal bimodal histogram, but it takes a lot of computation to determine threshold, especially multithreshold [3].

3. A Novel Imperialist Competitive Algorithm for Multithreshold Image Segmentation

ICA is a population-based metaheuristic. Each individual of population represents a country and some best countries are selected as imperialists in the initialization. Its main process is described as follows:

Step 1 (initialize the empire). An initial population P is generated randomly. Choose N_{imp} solutions with smallest cost as imperialists, and assign N_{col} remaining countries to the imperialists.

Step 2 (assimilation and revolution). In every empire, assimilation is carried out on every colony and executes revolution of some colonies.

Step 3. Exchange position of colony and imperialist if possible.

Step 4 (imperialist competition). According to the total cost of the empire, the weakest colonies of the worst empire are redistributed to the winning empire.

Step 5. If there are no members in an Empire, the Empire is eliminated directly.

Step 6. If the termination condition holds, the search ends; otherwise, go to Step 2.

The cost is related to the objective function. The better the objective function, the better the country. The specific process of the initial empire, assimilation, revolution, and imperialist empire competition of basic ICA can be found in Reference [21].

In the traditional imperialist competition, the new colony solution comes from the global search and local search. The global search or local search of imperialists is seldom considered. Imperialists are often excellent individuals in the population. The quality of solutions is higher than other colonies. Using these excellent individuals to search will help to improve the search efficiency and get better quality solutions more easily. For this reason, imperialists self-learning and reserve country set are introduced. In addition, in order to improve the search efficiency of the algorithm, a new strategy of revolution and assimilation is adopted. Above all, a novel ICA is proposed to solve multithreshold image segmentation. The steps of ICA are described in detail below.

Multithreshold image segmentation (assuming M level) is represented by an integer string $\phi = [\phi_1, \phi_2, \ldots, \phi_{M-1}]$. The decoding process is a series of integers arranged $M-1$ in ascending order $\overline{\phi} = [\overline{\phi_1}, \overline{\phi_2}, \ldots, \overline{\phi_{M-1}}]$, in which $\overline{\phi_i}$ is the order i segmentation threshold.

3.1. Initial Empire. The establishment of the initial empire is a key step for ICA. The specific steps are described as follows:

Step 1. Calculate the cost of country l $cost_l = f$, in which f is shown as the objective function value according to formula (2) or formula (3).

Step 2. Descending order for all solutions according to the cost of the country, choose the former N_{imp} countries as imperialists and the rest as the colony, so the number of colonies can be described as $N_{col} = N - N_{imp}$.

Step 3. Calculate the power of each imperialist k, $P_k = |cost_k / \sum_{i=1}^{N_{imp}} cost_i|$, and determine the colony number NC_l allocated to each imperialist k, where $NC_k = round\{P_k \times N_{col}\}$. The specific allocation steps are shown in Reference [21].

3.2. Assimilation. The assimilation is a process that the colonies in the empire gradually move close to their imperialists. It will make the colonies similar to their imperialists. When the colony is the same as their imperialist, the assimilation operation will lose its effect and reduce the diversity of the population. Therefore, in order to improve the efficiency of assimilation and the diversity of the population, a new strategy of assimilation is given.

Suppose that W_k is the set of all colonies in empire k.

The detailed steps of ICA are shown below.

Step 1. Suppose colony $\tau \in W_k$, and compute the consistency between colony τ and their imperialist k. If the two solutions are consistent, judge whether the reserve country set Ω is empty. If it is empty, the colony will not be assimilated. Otherwise, randomly choose a reserve country λ instead of τ.

Step 2. If colony τ is inconsistent with the imperialist k or has been replaced by the reserve country λ, the colony will move close to imperialist k through the global search operation.

The global search in Step 2 is divided into two steps. First, suppose the imperialist k is $\phi^k = [\phi_1^k, \phi_2^k, \ldots, \phi_{M-1}^k]$ and a colony τ^* is $\phi^\tau = [\phi_1^\tau, \phi_2^\tau, \ldots, \phi_{M-1}^\tau]$; then the new colony $\phi^{\tau^*} = [\phi_1^{\tau^*}, \phi_2^{\tau^*}, \ldots, \phi_{M-1}^{\tau^*}]$ can be obtained according to formula (4).

$$\phi_i^{\tau^*} = round\left(\phi_i^\tau + \beta \times \left(\phi_i^k - \phi_i^\tau\right)\right) \tag{4}$$

Among them, β is a random number between 0 and 2. And the colony τ^* can be amended and a feasible solution can be adjusted depending on formula (5).

$$\phi_i^{\tau^*} = \begin{cases} 0 & \phi_i^{\tau^*} < 0 \\ L-1 & \phi_i^{\tau^*} \geq L \\ \phi_i^{\tau^*} & 0 \leq \phi_i^{\tau^*} < L \end{cases} \tag{5}$$

The revised colony ϕ^{τ^*} is assimilated colony and replaced colony τ with colony ϕ^{τ^*}.

3.3. Imperialist Self-Learning. Imperialist self-learning and reserve country set are introduced to strengthen the development of imperialists and retain some suboptimal individuals. The algorithm search efficiency can be improved by the local search of imperialist. The suboptimal individuals obtained in the process of imperialist self-learning can be retained in the reserve country set. Therefore, the excellent individuals can be made better use.

For each imperialist k, $\phi^k = [\phi_1^k, \phi_2^k, \ldots, \phi_K^k]$ new individuals ϕ^{k^*} are obtained from the imperialist self-learning.

The specific imperialist self-learning process is shown as follows:

Step 1. Let $\phi^{k^*} = \phi^k$, generate a random integer α, and $\alpha \in [1, M-1]$.

Step 2. Take $\phi_\alpha^{k^*} = \phi_\alpha^k + rand \times L$, and $rand$ is the random number between 0 and 1.

Step 3. Use the formula (6) to modify $\phi_\alpha^{k^*}$, to get the new individual ϕ^{k^*}, and to calculate its cost value.

Step 4. If the cost value ϕ^{k^*} is better than ϕ^k, ϕ^k will be put into the reserve country set Ω and replaced ϕ^k; otherwise, ϕ^{k^*} will be put into the reserve country set.

$$\phi_\alpha^{k^*} = \begin{cases} 0 & \phi_\alpha^{k^*} < 0 \\ L-1 & \phi_\alpha^{k^*} \geq L \\ \phi_i^{\tau^*} & 0 \leq \phi_\alpha^{k^*} < L \end{cases} \tag{6}$$

Among them, the size of the reserve country set Ω is $3N_{imp}$, and if the number of reserve countries is larger than $3N_{imp}$, the weaker individuals will be eliminated according to the cost value.

3.4. Revolution. Revolution is another way for ICA to generate new solutions. A new way of revolution is proposed in this paper to balance the local search ability and global search ability and adopted different search strategies according to the problem characteristics.

Neighborhood search for the better individuals in the population can often improve the quality of the solution [26]. However, for the weaker individuals in the population, it is often difficult to obtain the better individuals by neighborhood search. Therefore, it is considered to select a reserve country randomly to eliminate the weaker individuals in the population in order to improve the overall quality of the population. The process of the revolution in each colony τ is described as follows.

Step 1. Generate a random number $rand$, if $rand < p_r$ continues; otherwise, end the revolution.

Step 2. If the cost value τ is in the first 80% of all colonies, then the new solution τ^* will directly replace the original colony τ after carrying out the consistent operation of the imperialist self-learning.

Step 3. If the cost value is in the worst 20% of all colonies, a reserve country from Ω will be chosen randomly to replace the original colony τ.

3.5. Imperialist Competition. Imperialist competition is the process of colonies redistribution. The weakest empire needs to hand over the weakest colonies to the winning empire, but the weakest colonies of the weakest empire often contribute less to the evolution of the winning empire and will weaken the competitiveness of the winning empire. Therefore, this paper considers randomly selecting a reserve country from Ω and compares with the weakest colonies of the weakest empire. Then, the better individual will be handed to the winning empire and the other is eliminated.

The specific operation of imperialist competition is described below.

Firstly, the empire total cost TC_k is calculated according to formula (7), which is related to the cost value of the imperialist k and all the empire colonies.

$$TC_k = cost_k + \xi \frac{\sum_{k=1}^{NC_l} cost_l}{NC_k} \tag{7}$$

Among them, $cost_l$ is the cost value of the colony l, and $cost_k$ is the cost value of the imperialist k. Weight coefficient ξ is less than 1, and it is 0.1 according to Reference [21]. Calculate the empire power $TP_k = TC_k / \sum_{i=1}^{N_{imp}} TC_i$. And the probability vector can be constructed as $D = TP - R = [TP_1 - r_1, TP_2 - r_2, \dots, TP_{N_{imp}} - r_{N_{imp}}]$. There, R is a random number vector between 0 and 1 and N_{im} is the number of imperialist. Suppose the value of element k in D is the largest; then the empire k is the winning empire. A reserve country x is randomly selected from Ω and compared x with the weakest colony x^* in the weakest empire. If x is better than x^*, x^* is replaced by x and x^* is allocated to the winning empire. Otherwise, x^* will be allocated directly to the winning empire.

3.6. Algorithm Flow. The flow chart of the hybrid imperialist competition algorithm is described in Figure 1. The termination condition is that the evaluation number of the objective function reaches a set value max_it.

4. Computational Experiments

In order to verify the superiority of the novel imperialist competition algorithm in solving multithreshold segmentation, a lot of computational experiments have been carried out. These experiments' programming has been implemented and run on a computer with 16.0G RAM and 2.80 GHz by using MATLAB 2015b.

4.1. Selection of Test Examples and Comparison Algorithm. In this paper, 10 standard test pictures are selected for the experiment. Ten pictures are named Test1-Test10. Test1-Test4 is from USC-SIPI with the picture size 512×512. Test5-Test10 is from BSD and the size of picture Test5 is 1024×1024. The size of Test6 and Test9-10 is 321×481. The size of Test7-8 is 481×321. They are shown in Figure 2.

In order to verify the effectiveness of the new ICA, we compare ICA with particle swarm optimization (PSO [13]), Gray Wolf algorithm (GWO [3]), and teaching optimization algorithm (TLBO [14]), which is proposed in recent years to solve multithreshold segmentation problems.

4.2. Parameters Settings. Four important parameters must be tested in new ICA: population size N, the ratio of number of imperialists to size N_{imp}/N, revolutionary probability p_m, and the maximum objective functions evaluation number max_it.

In order to study the influence of parameter selection on the algorithm performance and to find the best combination of parameters, a four-level DOE experiment is designed using Test1 test picture. The parameters of each level are shown in Table 1. According to the orthogonal Table 2, each group's parameters are run independently 10 times and the whole objective value is calculated. The results are shown in Table 2. According to the results obtained in Table 2, the influence of each parameter on the algorithm performance is ranked and it is shown in Table 3. The influence trend of each parameter on the performance of the algorithm is shown in Figure 3.

From Table 3, it can be seen that the ICA is mainly affected by the objective function evaluations maximum number and the population size. Thus, four parameters of the new imperial competition are as follows: $N = 60$, $N_{imp}/N = 15\%$, $P_r = 0.4$, and $max\_it = 8000$.

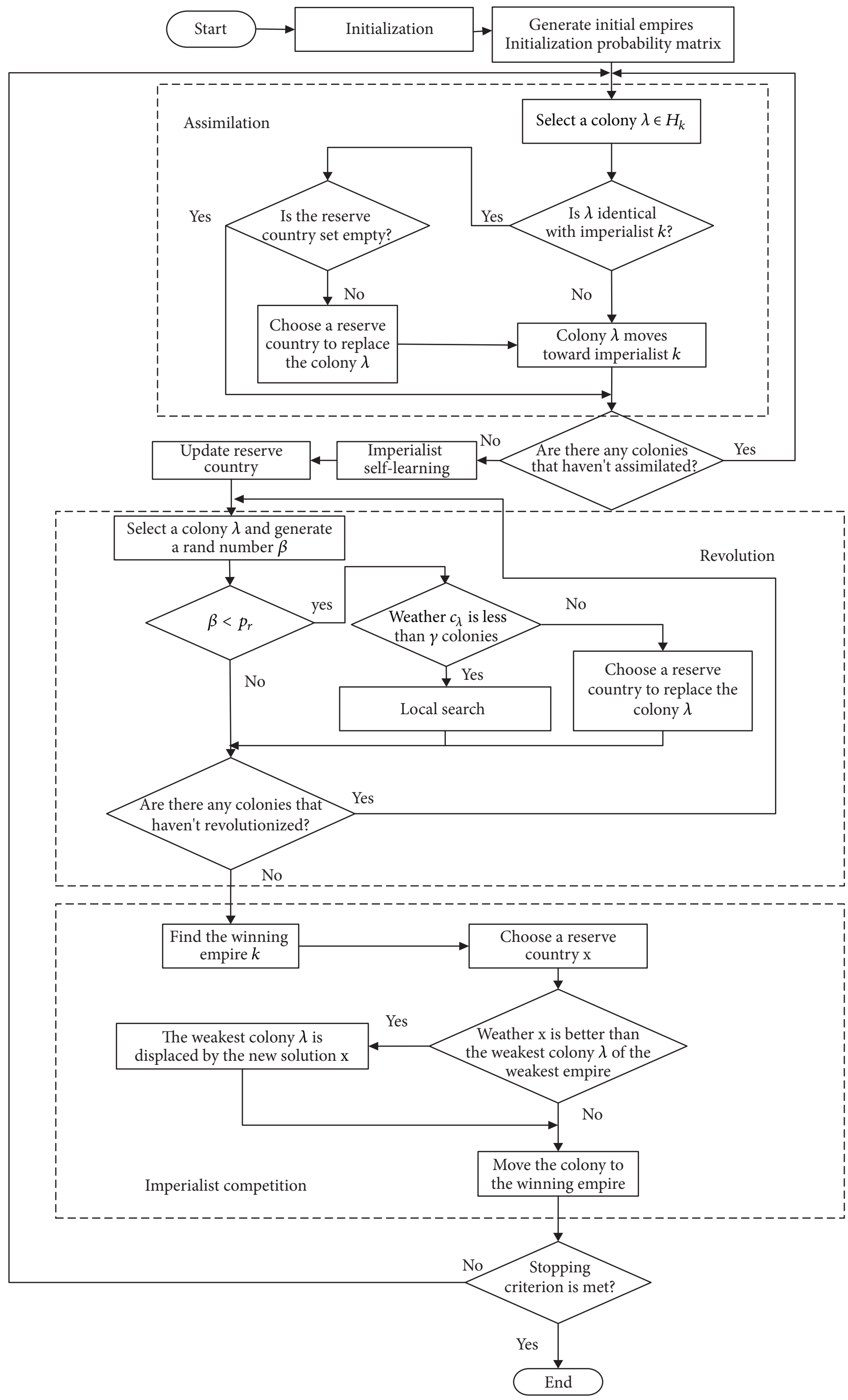
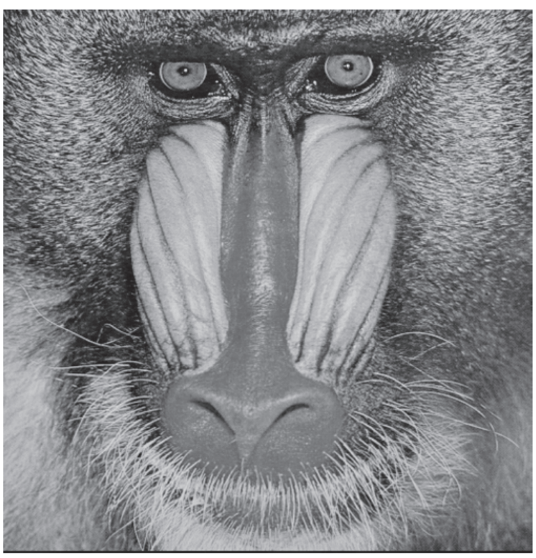


Figure 1: Algorithm flow chart.

4.3. Comparative Analysis with Other Algorithms. In order to analyze the superiority of the new ICA in solving the multithreshold segmentation problem, three algorithms, PSO [13], GWO [3], TLBO [14], are selected as the comparison algorithms. The termination conditions of the three algorithms are set to 8000, and the other parameters are set according to References [3, 13, 14]. Tables 4 and 5 give the average values and maximum values of Kapur's objective

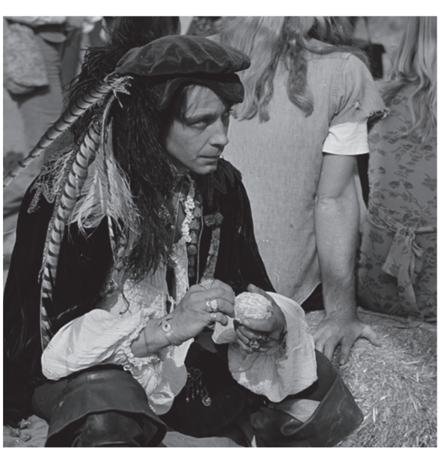
(a) Test1

(b) Test2



(c) Test3

(d) Test4



(e) Test5

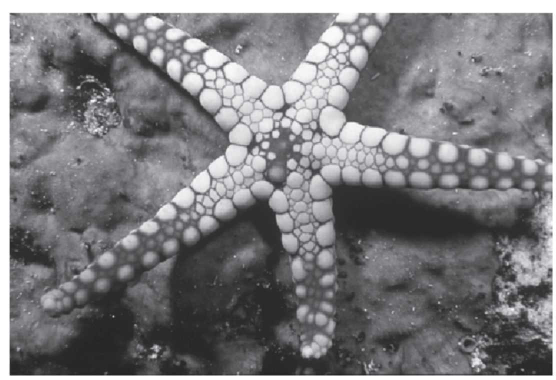
(f) Test6

Figure 2: Continued.

(g) Test7

(h) Test8

(i) Test9



(j) Test10

Figure 2: Test example.

values obtained by the new ICA and three contrast algorithms on 40 groups of images with four kinds of levels 3-6. Tables 6 and 7 show the average values and maximum values of Ostu's objective values obtained by four algorithms on 40 sets of examples. Tables 8 and 9 give the threshold values obtained by ICA, PSO, GWO, and TLBO based on Kapur's entropy methods and Ostu's between-class variance methods. $Testi\_M-1$ means the results of $Testi$ in M level. Figures 4 and 5 show the segmented images obtained by ICA-Kaur multilevel thresholding method and ICA-Ostu multilevel thresholding method. Table 10 shows the results of statistical analysis. We set confidence level as 95%. When $P\_\text{value} \leq 0.05$, the difference between algorithms is significant. Table 11 gives the average CPU time of all algorithms.

Table 1: Parameter values at various levels.

parameters	level			
	1	2	3	4
population size N	20	40	60	80
N_{imp}/N	5%	10%	15%	20%
revolutionary probability p_m	0.2	0.4	0.6	0.8
max_it	2000	4000	8000	10000

Table 2: Parameter orthogonal table and object function values.

Combination number	level				object function values
	N	N_{imp}/N	p_r	max_it	
1	1	1	1	1	17.9845
2	1	2	2	2	18.0123
3	1	3	3	3	18.0123
4	1	4	4	4	17.9946
5	2	1	2	3	17.9985
6	2	2	1	4	17.9985
7	2	3	4	1	17.9765
8	2	4	3	2	17.9865
9	3	1	3	4	18.0118
10	3	2	4	3	18.0083
11	3	3	1	2	18.0123
12	3	4	2	1	18.0006
13	4	1	4	2	17.9900
14	4	2	3	1	17.9908
15	4	3	2	4	18.0123
16	4	4	1	3	18.0118

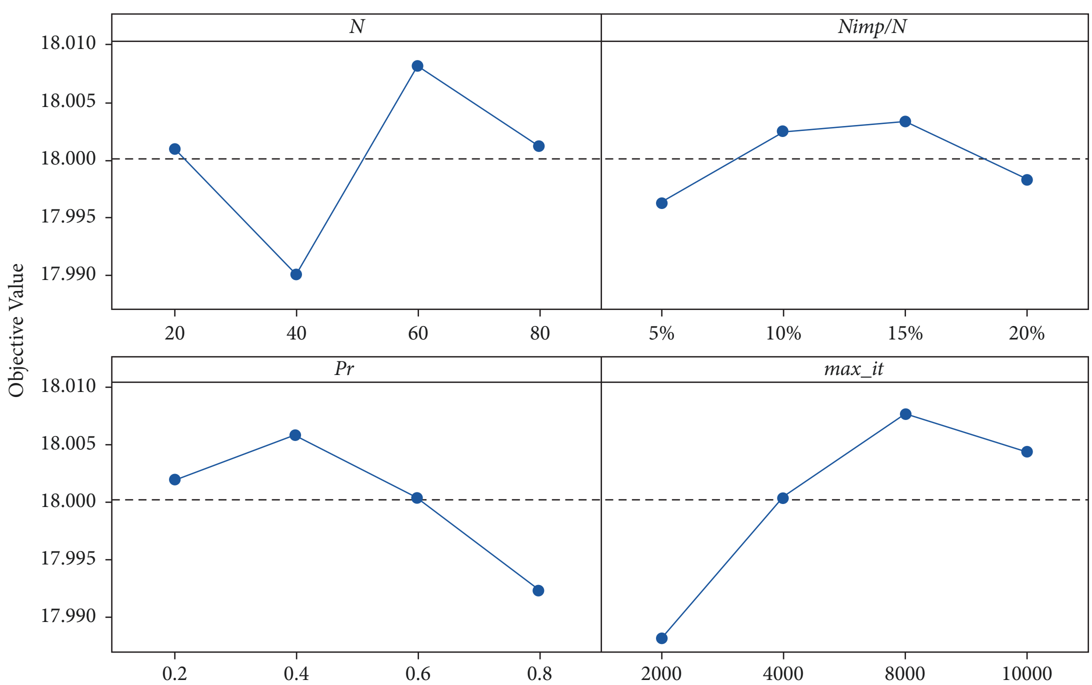


Figure 3: Main effect diagram of mean value.

Table 3: Average of parameters.

Level	N	N_{imp}/N	p_r	max_it
1	18.00	18.00	18.00	17.99
2	17.99	18.00	18.01	18.00
3	18.01	18.00	18.00	18.01
4	18.00	18.00	17.99	18.00
Delta	0.02	0.01	0.01	0.02
Rank	2	4	3	1

Firstly, compared with the four algorithms in solving Kapur's objective value, ICA has obtained the optimal solution in all 40 groups of examples, of which 30 groups are better than GWO and 19 groups are better than TLBO. Aiming at the average solution of 20 runs, among 40 groups of instances, the average solution of ICA has achieved better or equal to the results of the other four algorithms. Among them, 11 groups of instances are better than PSO, 30 groups are better than TLBO, and all instances are better than GWO. This also proves the stability of ICA algorithm. Thus ICA

Table 4: The maximum values of Kapur's objective values.

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test1_2	**12.3466**	12.3466	12.3463	12.3466	Test6_2	**12.6616**	12.6616	12.6616	12.6616
3	**15.3183**	15.3183	15.3183	15.3183	3	**15.8436**	15.8436	15.8431	15.8436
4	**18.0123**	18.0123	17.9996	18.0109	4	**18.6570**	18.6570	18.6481	18.6545
5	**20.6095**	20.6095	20.6008	20.6078	5	**21.4315**	21.4315	21.3996	21.4295
Test2_2	**12.2115**	12.2115	12.2115	12.2115	Test7_2	**12.9606**	12.9606	12.9605	12.9606
3	**15.5039**	15.5039	15.4997	15.5039	3	**16.2204**	16.2204	16.2198	16.2204
4	**18.3121**	18.3121	18.3072	18.3110	4	**19.3650**	19.3650	19.3181	19.3645
5	**20.9088**	20.9088	20.4932	20.9088	5	**22.1924**	22.1924	21.9295	22.1899
Test3_2	**12.2178**	12.2178	12.2178	12.2178	Test8_2	**11.2873**	11.2873	11.2873	11.2873
3	**15.2792**	15.2792	15.2750	15.2792	3	**14.1185**	14.1185	14.1148	14.1185
4	**18.1267**	18.1267	18.1216	18.1267	4	**16.7098**	16.7098	16.7087	16.7090
5	**20.7896**	20.7896	20.7226	20.7890	5	**19.7066**	19.7066	19.5136	19.7055
Test4_2	**12.6346**	12.6346	12.6346	12.6346	Test9_2	**12.6390**	12.6390	12.6390	12.6390
3	**15.6887**	15.6887	15.6887	15.6887	3	**15.5952**	15.5952	15.5952	15.5950
4	**18.5394**	18.5394	18.4917	18.5387	4	**18.5295**	18.5295	18.5183	18.5292
5	**21.2818**	21.2818	21.0860	21.2746	5	**21.1922**	21.1922	21.1899	21.1853
Test5_2	**12.6357**	12.6357	12.6355	12.6357	Test10_2	**12.9682**	12.9682	12.9682	12.9682
3	**15.8115**	15.8115	15.8075	15.8115	3	**16.1254**	16.1254	16.1145	16.1254
4	**18.7441**	18.7441	18.6589	18.7362	4	**19.0579**	19.0579	19.0300	19.0576
5	**21.6195**	21.6195	21.5462	21.6057	5	**21.8090**	21.8090	21.8043	21.8050

Table 5: The average values of Kapur's objective values.

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test1_2	**12.3466**	12.3466	12.2945	12.3466	Test6_2	**12.6616**	12.6616	12.6167	12.6616
3	**15.3183**	15.3183	15.2436	15.3181	3	**15.8436**	15.8436	15.7336	15.8434
4	**18.0095**	18.0057	17.8894	18.0023	4	**18.6512**	18.6436	18.3885	18.6363
5	**20.6095**	20.6095	20.1170	20.5508	5	**21.4315**	21.4315	20.8351	21.4053
Test2_2	**12.2115**	12.2115	12.1910	12.2114	Test7_2	**12.9606**	12.9606	12.9265	12.9606
3	**15.5039**	15.5039	15.2467	15.5013	3	**16.2204**	16.2204	16.0922	16.2178
4	**18.3121**	18.3121	18.0079	18.3087	4	**19.3650**	19.3650	19.0597	19.3487
5	**20.9087**	20.9087	20.0152	20.8759	5	**22.1924**	22.1393	21.5222	22.1388
Test3_2	**12.2178**	12.2178	12.1838	12.2178	Test8_2	**11.2873**	11.2873	11.2329	11.2873
3	**15.2792**	15.2792	15.1968	15.2790	3	14.1184	**14.1185**	13.9991	14.1177
4	**18.1267**	18.1264	17.9916	18.1130	4	**16.6985**	16.6646	16.5734	16.6814
5	**20.7896**	20.7895	20.3838	20.7483	5	**19.7066**	19.6472	19.1285	19.6403
Test4_2	**12.6346**	12.6346	12.5996	12.6346	Test9_2	**12.6390**	12.6390	12.6015	12.6390
3	**15.6887**	15.6887	15.6841	15.6877	3	**15.5952**	15.5952	15.5491	15.5936
4	18.5221	18.5066	18.3674	**18.5258**	4	**18.5295**	18.5295	18.3060	18.5248
5	**21.2818**	21.2626	20.6440	21.2553	5	21.1794	**21.1922**	20.8378	21.1554
Test5_2	**12.6357**	12.6357	12.5795	12.6357	Test10_2	**12.9682**	12.9682	12.9528	12.9682
3	**15.8115**	15.8115	15.6541	15.8107	3	**16.1254**	16.1254	15.9787	16.1251
4	**18.7347**	18.6983	18.4361	18.7110	4	**19.0579**	19.0579	18.7716	19.0463
5	**21.6195**	21.5351	21.1728	21.5724	5	21.8082	**21.8088**	21.3168	21.7935

Table 6: The maximum values of Ostu's objective values.

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test1_2	**1961.817**	1961.817	1961.790	1961.817	Test6_2	**2976.191**	2976.191	2976.191	2976.191
3	**2128.308**	2128.308	2128.308	2128.308	3	**3145.463**	3145.463	3145.463	3145.463
4	**2191.870**	2191.870	2190.602	2191.870	4	**3216.403**	3216.403	3215.399	3216.271
5	**2217.799**	2217.799	2209.967	2217.726	5	**3255.995**	3255.995	3238.241	3255.908
Test2_2	**1948.719**	1948.719	1948.719	1948.719	Test7_2	**7209.906**	7209.906	7208.633	7209.906
3	**2024.834**	2024.834	2024.834	2024.834	3	**7531.419**	7531.419	7531.235	7531.419
4	**2070.077**	2070.077	2066.278	2070.057	4	**7617.718**	7617.718	7602.299	7617.682
5	**2096.139**	2096.139	2075.997	2095.610	5	**7666.598**	7666.598	7662.395	7666.079
Test3_2	**1549.083**	1549.083	1549.052	1549.083	Test8_2	**5068.434**	5068.434	5068.434	5068.434
3	**1639.532**	1639.532	1639.532	1639.532	3	**5232.919**	5232.919	5232.833	5232.919
4	**1693.195**	1693.195	1692.378	1693.195	4	**5310.356**	5310.356	5309.691	5310.265
5	**1719.057**	1719.057	1718.145	1718.883	5	**5349.222**	5349.222	5346.560	5349.043
Test4_2	**2532.321**	2532.321	2532.321	2532.321	Test9_2	**1665.411**	1665.411	1665.411	1665.411
3	**2703.572**	2703.572	2703.374	2703.572	3	**1746.197**	1746.197	1745.604	1746.197
4	**2766.459**	2766.459	2765.768	2766.371	4	**1797.124**	1797.124	1797.098	1797.015
5	**2810.842**	2810.842	2809.046	2810.694	5	**1823.625**	1823.625	1821.739	1823.002
Test5_2	**2947.502**	2947.502	2947.502	2947.502	Test10_2	**2551.975**	2551.975	2551.975	2551.975
3	**3126.855**	3126.855	3126.467	3126.855	3	**2784.227**	2784.227	2784.227	2784.227
4	**3208.810**	3208.810	3208.694	3208.694	4	**2869.431**	2869.431	2869.415	2869.431
5	**3254.797**	3254.797	3248.700	3254.562	5	**2916.273**	2916.273	2915.191	2916.087

Table 7: The average values of Ostu's objective values.

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test1_2	**1961.817**	1961.817	1938.178	1961.812	Test6_2	**2976.191**	2976.191	2953.020	2976.191
3	**2128.308**	2128.308	2090.090	2128.195	3	**3145.463**	3145.463	3095.006	3145.392
4	**2191.870**	2191.870	2154.799	2191.482	4	**3216.403**	3216.403	3163.828	3215.248
5	2217.231	**2217.516**	2178.598	2216.291	5	***3255.995***	3253.210	3209.790	3253.837
Test2_2	**1948.719**	1948.719	1943.368	1948.719	Test7_2	**7209.906**	7209.906	7181.118	7209.906
3	2024.831	**2024.834**	2015.234	2024.772	3	**7531.419**	7531.419	7414.411	7531.202
4	***2070.077***	2070.076	2045.085	2069.334	4	**7617.718**	7617.718	7556.210	7614.405
5	***2096.139***	2096.135	2062.247	2094.870	5	***7665.778***	7664.147	7621.531	7662.482
Test3_2	**1549.083**	1549.083	1518.499	1549.083	Test8_2	**5068.434**	5068.434	5033.695	5068.434
3	**1639.532**	1639.532	1620.771	1639.385	3	**5232.919**	5232.919	5209.283	5232.787
4	***1693.195***	1693.190	1647.427	1692.546	4	**5310.356**	5310.356	5273.633	5309.334
5	1719.041	**1719.044**	1694.932	1717.250	5	***5349.222***	5349.221	5317.495	5347.035
Test4_2	**2532.321**	2532.321	2519.423	2532.312	Test9_2	**1665.411**	1665.411	1657.895	1665.411
3	**2703.572**	2703.572	2667.182	2703.274	3	**1746.197**	1746.197	1739.602	1746.118
4	**2766.459**	2766.459	2704.149	2765.460	4	1797.117	**1797.124**	1778.585	1796.539
5	***2810.842***	2810.838	2767.802	2809.198	5	1823.590	**1823.618**	1796.961	1821.105
Test5_2	**2947.502**	2947.502	2905.698	2947.500	Test10_2	**2551.975**	2551.975	2512.428	2551.975
3	**3126.855**	3126.855	3093.664	3126.821	3	**2784.227**	2784.227	2749.713	2783.919
4	**3208.810**	3208.810	3177.217	3207.818	4	2869.428	**2869.431**	2827.326	2868.631
5	3254.775	**3254.797**	3223.476	3251.292	5	***2916.271***	2916.270	2870.529	2913.415

Table 8: The threshold values obtained by ICA, PSO, GWO, and TLBO based on Kapur's entropy methods.

Example	ICA	PSO	GWO	TLBO
Test1_2	97 164	97 164	98 164	97 164
3	82 126 175	82 126 175	82 126 175	82 126 175
4	64 97 137 179	64 97 137 179	62 97 140 178	64 97 138 180
5	63 94 128 163 194	63 94 128 163 194	62 94 129 161 194	62 93 126 162 193
Test2_2	71 173	71 173	71 173	71 173
3	69 127 183	69 127 183	67 127 183	69 127 183
4	67 106 145 185	67 106 145 185	67 104 146 185	66 105 144 185
5	60 89 123 155 187	60 89 123 155 187	44 90 125 142 183	60 89 123 155 187
Test3_2	78 143	78 143	18 143	78 143
3	45 98 152	45 98 152	52 103 154	45 98 152
4	33 73 114 159	33 73 114 159	38 78 118 161	33 73 114 159
5	33 68 103 138 172	33 68 103 138 172	28 54 94 134 171	33 68 103 137 171
Test4_2	75 147	75 147	75 147	75 147
3	61 113 165	61 113 165	61 113 165	61 113 165
4	58 105 148 194	58 105 148 194	62 100 149 196	57 105 149 194
5	42 77 114 154 195	42 77 114 154 195	35 83 133 164 194	40 76 116 155 195
Test5_2	91 172	91 172	90 172	91 172
3	62 117 174	62 117 174	64 115 174	62 117 174
4	62 117 174 230	62 117 174 230	50 100 136 176	59 115 173 230
5	46 89 131 174 230	46 89 131 174 230	36 88 131 172 231	42 90 134 175 230
Test6_2	99 172	99 172	99 172	99 172
3	88 138 194	88 138 194	87 138 194	88 138 194
4	87 135 174 214	87 135 174 214	89 132 172 214	88 134 173 214
5	63 98 136 174 214	63 98 136 174 214	62 95 139 172 212	63 98 137 174 215
Test7_2	94 168	94 168	94 171	94 168
3	53 110 164	53 110 164	52 110 164	53 110 164
4	53 110 163 212	53 110 163 212	54 104 165 211	55 110 163 212
5	38 78 118 163 212	38 78 118 163 212	46 99 138 175 207	38 76 118 163 213
Test8_2	48 104	48 104	48 104	48 104
3	36 72 115	36 72 115	41 75 116	36 72 115
4	35 69 102 135	35 69 102 135	36 69 102 135	35 69 102 134
5	46 101 153 201 243	46 101 153 201 243	36 72 147 192 244	46 102 153 202 242
Test9_2	115 181	115 181	115 181	115 181
3	112 158 202	112 158 202	112 158 202	74 119 183
4	73 116 161 205	73 116 161 205	75 116 156 200	74 116 161 205
5	73 114 150 185 220	73 114 150 185 220	71 112 149 183 220	71 115 150 184 221
Test10_2	91 170	91 170	91 170	91 170
3	75 130 183	75 130 183	80 130 183	75 130 183
4	68 116 164 206	68 116 164 206	73 113 166 208	67 116 164 206
5	56 93 132 170 209	56 93 132 170 209	55 95 134 174 212	54 90 129 170 209

can get better results than other three algorithms on most of instances in similar computation times and has advantages in solving multithreshold image segmentation.

For Ostu's objective value, ICA also obtained all the best solutions in 40 groups of examples, and 28 groups of examples were better than GWO and 17 groups of examples were better than TLBO, which further verified the high quality of ICA solution. In terms of the average solution of 20 runs, ICA achieved better or equal results to the other four algorithms in 33 groups of instances, and 8 groups of instances were better than PSO, 33 groups were better than TLBO, and 40 groups were better than GWO. Furthermore, the stability of ICA solution is verified again. The statistical results in Table 10 also validate this conclusion, which proves the superiority of ICA in solving multithreshold segmentation.

5. Conclusion

A new imperialist competition algorithm is proposed in order to solve the multithreshold segmentation problem

Table 9: The threshold values obtained by ICA, PSO, GWO, and TLBO based on Ostu's between-class variance methods.

Example	ICA	PSO	GWO	TLBO
Test1_2	93 151	93 151	92 151	93 151
3	81 127 171	81 127 171	81 127 171	81 127 171
4	75 114 145 180	75 114 145 180	75 111 142 181	75 113 145 180
5	73 109 136 160 188	73 109 136 160 188	70 86 119 150 180	74 110 137 160 188
Test2_2	113 173	113 173	113 173	113 173
3	93 145 191	93 145 191	93 145 191	93 145 191
4	84 129 172 203	84 129 172 203	93 128 170 203	85 129 173 203
5	69 107 143 180 205	69 107 143 180 205	90 124 162 171 204	70 104 142 178 205
Test3_2	98 150	98 150	97 150	98 150
3	86 125 161	86 125 161	86 125 161	86 125 161
4	72 106 137 168	72 106 137 168	68 105 136 167	72 106 137 168
5	68 100 126 150 175	68 100 126 150 175	63 95 123 149 175	66 97 123 148 174
Test4_2	68 135	68 135	68 135	68 135
3	63 119 166	63 119 166	61 118 166	63 119 166
4	47 86 126 169	47 86 126 169	49 88 129 170	47 85 125 169
5	43 79 113 146 177	43 79 113 146 177	48 82 111 144 176	42 78 112 147 178
Test5_2	57 124	57 124	57 124	57 124
3	39 92 142	39 92 142	40 93 141	39 92 142
4	35 82 124 164	35 82 124 164	35 81 123 164	35 81 123 164
5	28 65 100 134 172	28 65 100 134 172	37 75 102 134 172	28 64 100 133 172
Test6_2	87 162	87 162	87 162	87 162
3	82 136 190	82 136 190	82 136 190	82 136 190
4	74 112 149 197	74 112 149 197	76 110 147 196	74 112 149 198
5	73 109 143 183 221	73 109 143 183 221	71 88 116 150 197	72 108 143 183 220
Test7_2	60 157	60 157	62 155	60 157
3	54 131 206	54 131 206	53 131 206	54 131 206
4	40 84 143 208	40 84 143 208	49 96 135 206	40 84 142 208
5	39 81 136 191 227	39 81 136 191 227	45 81 134 191 229	38 81 138 194 228
Test8_2	108 196	108 196	108 196	108 196
3	89 135 206	89 135 206	90 135 206	89 135 206
4	68 106 145 210	68 106 145 210	71 108 145 209	69 106 145 210
5	63 97 126 157 215	63 97 126 157 215	69 97 124 158 214	63 98 127 158 218
Test9_2	78 144	78 144	78 144	78 144
3	75 119 177	75 119 177	75 122 177	75119 177
4	66 90 131 186	66 90 131 186	65 89 131 186	66 90 132 188
5	59 79 101 140 191	59 79 101 140 191	61 82 106 151 192	59 78 100 141 195
Test10_2	85 157	85 157	85 157	85 157
3	69 120 178	69 120 178	69 120 178	69 120 178
4	60 101 138 187	60 101 138 187	61 101 138 187	60 101 138 187
5	52 86 117 150 194	52 86 117 150 194	53 86 114 147 194	52 87 118 150 195

Table 10: Statistical analysis results.

t-test	P_value (DI_R)	P_value (ρ)
t-test(ICA,PSO)	0.001	0.041
t-test(ICA,GWO)	≤ 0.001	≤ 0.001
t-test(ICA,TLBO)	≤ 0.001	≤ 0.001

in this paper. A new strategy of revolution, assimilation, imperialist self-learning, and a novel imperialist competition can be designed to improve the algorithm performance. Ten standard test pictures are selected to test and compared with three new optimization algorithms. The experimental results show that the proposed imperialist competition algorithm has fast convergence speed, high quality, and high stability in solving multithreshold segmentation problems.

Meanwhile, the algorithm in this paper will be less affected by noise, because the algorithm in this paper only pursues the optimization of the objective function. The goal of multithreshold image segmentation is to separate

Figure 4: Continued.

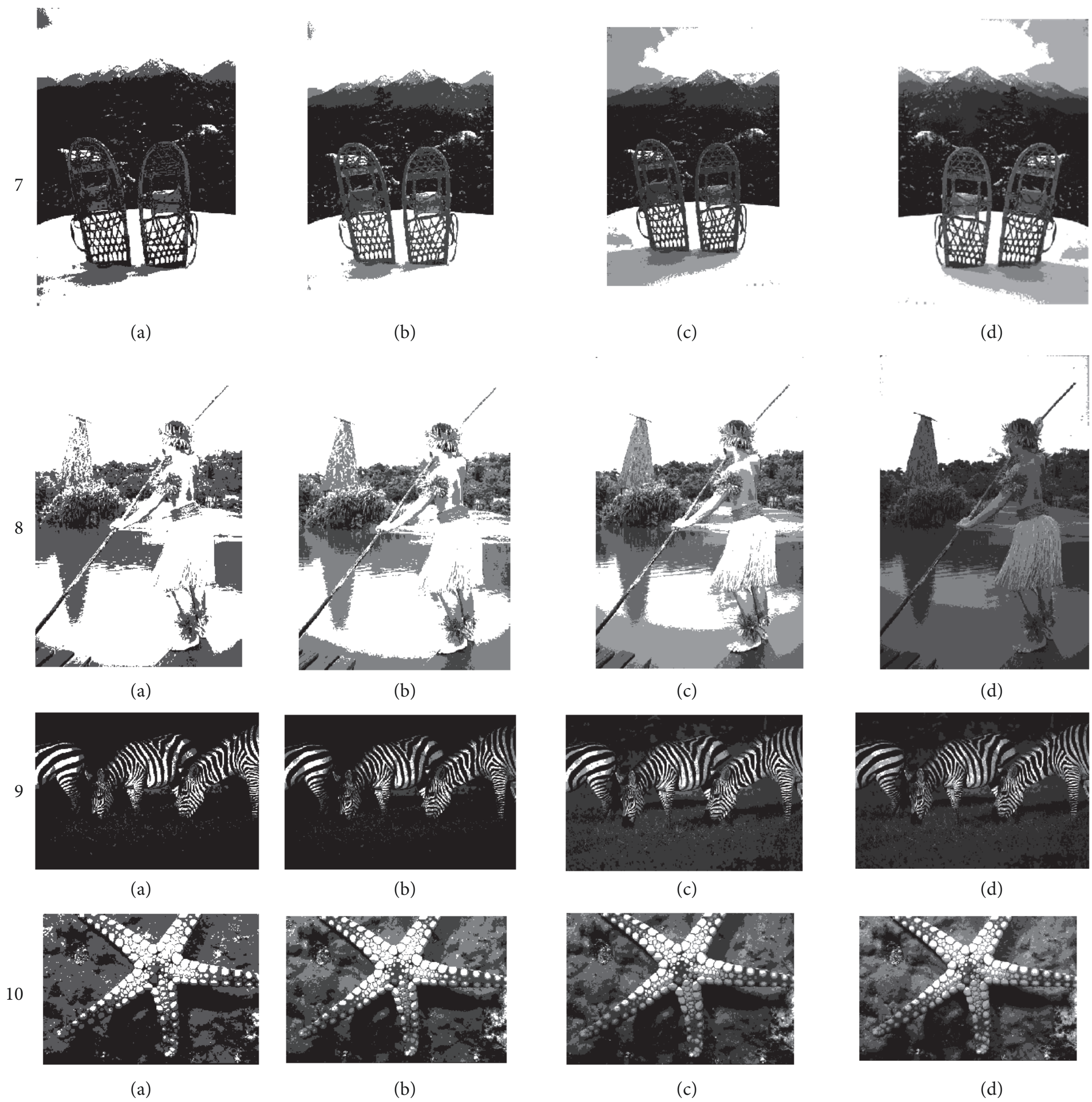


Figure 4: Segmented images obtained by ICA-Kapur multilevel thresholding method (a) represent 3-level thresholding, (b) represent 3-level thresholding, (c) represent 4-level thresholding, and (d) represent 6-level thresholding.

multiple objects of interest. The image noise is usually an isolated point. Thus, we can remove the noise points by mathematical morphological small targets after threshold target segmentation.

Although Kapur and Otsu methods are two widely used image threshold methods, they still cannot satisfy all kinds of images. Therefore, simply seeking the optimal value of one of the indicators is easy to make the segmentation result appear over and under segmentations, which is also the limitation of our algorithm in this paper. In the future, we will continue to consider adopting multiobjective evolutionary algorithm to achieve better optimization effect by balancing different indexes at the same time. This problem will be our next research step.

Data Availability

The data we used to support the findings of this study is available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

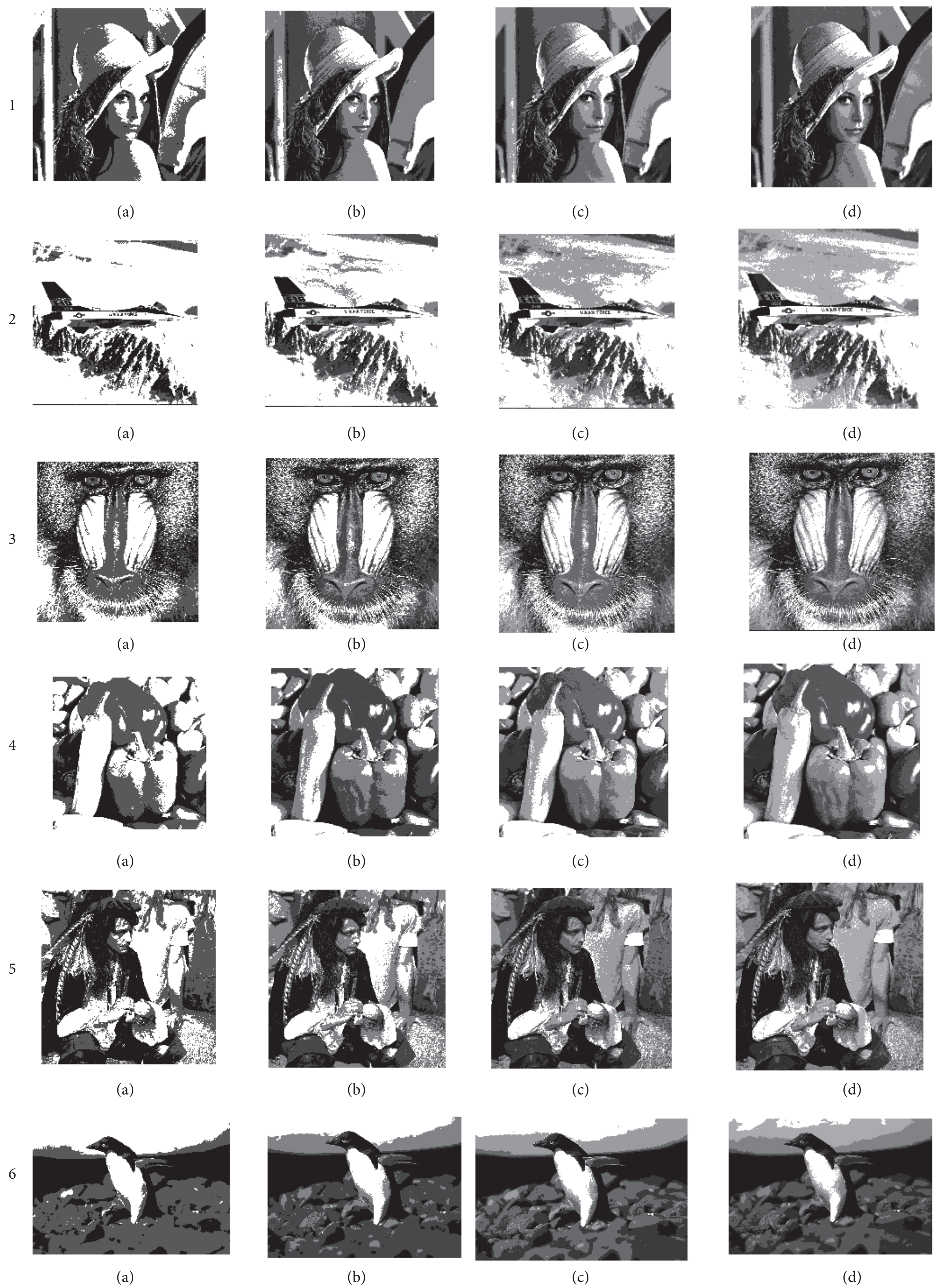


Figure 5: Continued.

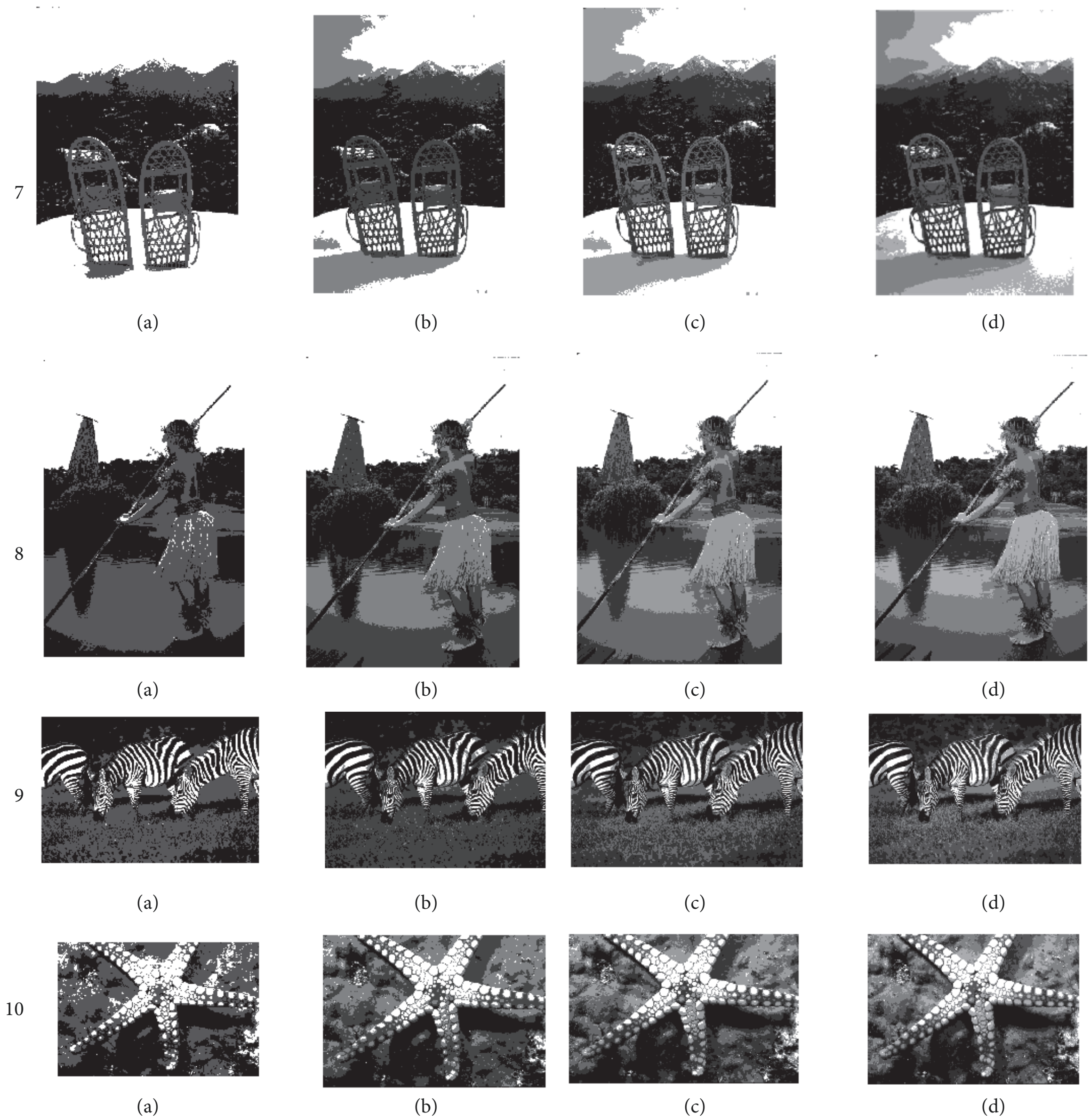


FIGURE 5: Segmented images obtained by ICA-Ostu multilevel thresholding method (a) represent 3-level thresholding, (b) represent 3-level thresholding, (c) represent 4-level thresholding, and (d) represent 6-level thresholding.

TABLE 11: The average CPU time of all algorithms (in seconds).

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test1_2	1.0148	1.8466	1.0573	1.2695	Test6_2	0.9013	1.8463	0.9634	1.1757
3	0.9968	1.9547	1.0824	1.2020	3	1.0516	2.0183	1.0631	1.4172
4	1.0817	2.1041	1.0777	1.2727	4	1.0725	2.1534	1.0789	1.3710
5	1.1677	2.2427	1.1135	1.3929	5	1.0992	2.2310	1.1540	1.4416
Test2_2	0.9160	2.1234	0.9569	1.1933	Test7_2	0.9479	1.9786	1.0088	1.2306
3	1.0238	2.1349	1.0966	1.2583	3	1.0616	2.0336	1.0571	1.3009
4	1.0121	2.2104	1.1203	1.4549	4	1.0868	2.2236	1.1175	1.3781
5	1.0935	2.2470	1.1195	1.4557	5	1.1588	2.3927	1.1682	1.4598

Table 11: Continued.

Example	ICA	PSO	GWO	TLBO	Example	ICA	PSO	GWO	TLBO
Test3_2	0.9684	1.8421	0.9593	1.2024	Test8_2	0.9654	1.9061	1.0042	1.2234
3	0.9690	1.9660	1.0271	1.2812	3	1.0587	2.0592	1.0648	1.2818
4	1.0981	2.1097	1.0739	1.3281	4	1.1558	2.2072	1.1166	1.3552
5	1.0956	2.2191	1.1637	1.4219	5	1.1965	2.3133	1.1706	1.4322
Test4_2	0.9394	1.8492	0.9677	1.2796	Test9_2	0.9599	1.8732	0.9750	1.2245
3	1.0026	1.9714	1.0213	1.2786	3	1.0476	1.9952	1.0556	1.2709
4	1.0562	2.1719	1.1014	1.3773	4	1.0656	2.1281	1.0893	1.3672
5	1.1098	2.2701	1.1188	1.3913	5	1.1195	2.2444	1.1403	1.4266
Test5_2	0.9514	2.3564	0.9939	1.2206	Test10_2	0.9563	1.9158	0.9922	1.3281
3	1.0401	2.0438	1.0557	1.2901	3	1.0342	2.0040	1.0496	1.3778
4	1.0839	2.1590	1.1043	1.3911	4	1.0759	2.2321	1.0948	1.3701
5	1.1561	2.3302	1.1637	1.4313	5	1.1585	2.3338	1.1481	1.4289

Acknowledgments

This work is financially supported by Shandong Province Science and Technology Development Plan Project Foundation (No. 2014GGX101030).

References

[1] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[2] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision Graphics and Image Processing*, vol. 29, no. 1, pp. 273–285, 1985.

[3] A. K. M. Khairuzzaman and S. Chaudhury, "Multilevel thresholding using grey wolf optimizer for image segmentation," *Expert Systems with Applications*, vol. 86, pp. 64–76, 2017.

[4] M. Ali, C. W. Ahn, and M. Pant, "Multi-level image thresholding by synergetic differential evolution," *Applied Soft Computing*, vol. 17, pp. 1–11, 2014.

[5] A. Alihodzic and M. Tuba, "Improved bat algorithm applied to multilevel image thresholding," *The Scientific World Journal*, vol. 2014, Article ID 176718, 16 pages, 2014.

[6] S. Arora, J. Acharya, A. Verma, and P. K. Panigrahi, "Multilevel thresholding for image segmentation through a fast statistical recursive algorithm," *Pattern Recognition Letters*, vol. 29, no. 2, pp. 119–125, 2008.

[7] A. K. Bhandari, V. K. Singh, A. Kumar, and G. K. Singh, "Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3538–3560, 2014.

[8] J. Brest, U. Mlakar, and B. Poto, "A hybrid differential evolution for optimal multilevel image thresholding," *Pergamon Press, Inc*, vol. 65, pp. 221–232, 2016.

[9] H. M. Mohamed and E. Mahmoud, "Efficient solution of OSTU multilevel image thresholding: a comparative study," *Expert Systems with Applications*, vol. 116, pp. 299–309, 2019.

[10] V. K. Bohat and K. Arya, "A new heuristic for multilevel thresholding of images," *Expert Systems with Applications*, vol. 117, pp. 176–203, 2019.

[11] E. Cuevas, V. Osuna-Enciso, D. Zaldivar et al., "Multithreshold segmentation based on artificial immune systems," *Mathematical Problems in Engineering*, vol. 2012, Article ID 874761, 20 pages, 2012.

[12] E. Cuevas, A. González, F. Fausto, D. Zaldívar et al., "Multithreshold segmentation by using an algorithm based on the behavior of locust swarms," *Mathematical Problems in Engineering*, vol. 2015, Article ID 805357, 25 pages, 2015.

[13] P.-Y. Yin, "Multilevel minimum cross entropy threshold selection based on particle swarm optimization," *Applied Mathematics and Computation*, vol. 184, no. 2, pp. 503–513, 2007.

[14] H. S. Gill, B. S. Khehra, A. Singh, and L. Kaur, "Teaching-learning based optimization algorithm to minimize cross entropy for selecting multilevel threshold values," *Egyptian Informatics Journal*, 2018.

[15] S. Pare, A. Kumar, V. Bajaj, and G. K. Singh, "Amultilevel color image segmentation technique based on cuckoo search algorithm and energy curve," *Applied Soft Computing*, vol. 47, pp. 76–102, 2016.

[16] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "A fast scheme for multilevel thresholding based on a modified bee algorithm," *Knowledge-Based Systems*, vol. 101, pp. 114–134, 2016.

[17] B. Akay, "A study on particle swarm optimization and artificial bee colony algoritthms for multilevel thresholding," *Applied Soft Computing*, vol. 13, no. 6, pp. 3066–3091, 2013.

[18] X. L. Zhang, T. Yang, and N. N. Cui, "Flame image segmentation based on the bee colony algorithm with characteristics of levy flights," *Mathematical Problems in Engineering*, vol. 2015, Article ID 805075, 8 pages, 2015.

[19] R. K. Sambandam and S. Jayaraman, "Self-adaptive dragonfly based optimal thresholding for multilevel segmentation of digital images," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 449–461, 2018.

[20] M. A. E. Aziz, A. A. Ewees, and A. E. Hassanien, "Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation," *Expert Systems with Applications*, vol. 83, pp. 242–256, 2017.

[21] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in *Proceedings of the Congress on Evolutionary Computation (CEC '07)*, pp. 4661–4667, IEEE, Singapore, September 2007.

[22] A. Rabiee, M. Sadeghi, and J. Aghaei, "Modified imperialist competitive algorithm for environmental constrained energy management of microgrids," *Journal of Cleaner Production*, vol. 202, pp. 273–292, 2018.

[23] R. Akbari, M. Abbasi, F. Faghihi, S. M. Mirvakili, and J. Mokhtari, "A novel multi-objective optimization method, imperialist competitive algorithm, for fuel loading pattern of nuclear reactors," *Progress in Nuclear Energy*, vol. 108, pp. 391–397, 2018.

[24] A. Fathy and H. Rezk, "Parameter estimation of photovoltaic system using imperialist competitive algorithm," *Journal of Renewable Energy*, vol. 111, pp. 307–320, 2017.

[25] D. M. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Transactions on Cybernetics*, pp. 1–13, 2018.

[26] Z. X. Pan, D. M. Lei, and Q. Y. Zhang, "A new imperialist competitive algorithm for multiobjective low carbon parallel machines scheduling," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5914360, 13 pages, 2018.

[27] P. Zhang, Y. Lv, and J. Zhang, "An improved imperialist competitive algorithm based photolithography machines scheduling," *International Journal of Production Research*, vol. 56, no. 3, pp. 1–13, 2017.

[28] S. Karimi, Z. Ardalan, B. Naderi, and M. Mohammadi, "Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 41, pp. 667–682, 2017.

[29] S. Hany, Z. Mostafa, F. Hamed, and M. Iraj, "Simulated imperialist competitive algorithm in two-stage assembly flow shop with machine breakdowns and preventive maintenance," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 230, no. 5, pp. 934–953, 2016.

[30] S. Hosseini and A. Al Khaled, "A survey on the Imperialist Competitive Algorithm metaheuristic: implementation in engineering domain and directions for future research," *Applied Soft Computing*, vol. 24, no. C, pp. 1078–1094, 2014.