scientific reports

Check for updates

OPEN

Multilevel thresholding of color images using globally informed artificial bee colony algorithm

Ivona Brajević[1]✉ & Jelena Ignjatović[2]

Multilevel image thresholding presents a computational challenge as the number of thresholds increases, requiring efficient optimization techniques. The artificial bee colony (ABC) algorithm is a widely used metaheuristic for addressing this problem. Despite the good performance of the ABC algorithm, it struggles with an inadequate balance between discovering new solutions and refining existing ones. This paper presents the globally informed artificial bee colony (giABC), an enhanced ABC variant, proposed for multilevel color image thresholding. To overcome the limitations of the ABC algorithm, giABC introduces two novel mutation operators. In the employed phase, solutions are dynamically guided toward the mean of the current better solutions, ensuring a sustained balance between global exploration and local enhancement. In the onlooker phase, solutions are further refined by combining attraction to the global best solution with adaptation to promising solutions, significantly enhancing both convergence speed and solution quality. The proposed giABC, along with the ABC, its two variants and the chaotically-enhanced Rao algorithm, were tested on twelve color images from the Berkeley dataset using Otsu's objective function. Experimental results show that giABC outperforms the other metaheuristics in accuracy, robustness, peak signal-to-noise ratio and structural similarity index, with Wilcoxon signed-rank tests confirming its statistical significance.

Keywords Multilevel color image thresholding, Otsu method, Artificial bee colony algorithm, Optimization

Image segmentation is a core task in image processing that involves dividing a digital image into multiple segments based on specific characteristics[1]. These regions are typically homogeneous in terms of specific properties like color or texture. The purpose of segmentation is to change the image representation to make it more meaningful to analyze. Image segmentation is widely used across various industries, including medical imaging[2], surveillance systems, robotics and autonomous vehicles[3].

Thresholding is among the most frequently used techniques for image segmentation because of its simplicity in implementation and effectiveness[4]. The core idea of thresholding is to classify the pixels of a colored or grayscale image into different regions based on their intensity values by setting specific threshold values. The main challenge is determining the correct thresholds. When an image is split into two regions, this process is known as bi-level thresholding. Bi-level thresholding is extendable to multilevel thresholding when the goal is to divide an image into more than two regions.

Over the years, a wide range of thresholding techniques has been introduced[5]. Most of these techniques rely on optimizing a specific criterion function. Maximum between-class variance and maximum entropy are two of the most extensively employed criteria for identifying optimal threshold values[6]. Otsu's between-class variance criterion selects the appropriate thresholds by maximizing the variance across regions. Entropy-based criteria, such as Kapur's entropy or Renyi's entropy, aim to maximize the sum of entropies for each region to determine the suitable thresholds[7].

Multilevel image thresholding involves finding k optimal integer thresholds between 0 and 255 and represents a task that belongs to the class of NP-hard combinatorial optimization problems[5]. The computational time of current deterministic algorithms for multilevel thresholding grows exponentially with the increase in the number of thresholds. Consequently, these methods are not practical for solving the problem within a realistic time limit[8].

Given the inefficiency of standard deterministic methods, researchers over the last few decades have increasingly adopted metaheuristic optimization algorithms for multilevel thresholding[9–11]. These algorithms do not ensure finding the optimal solution but effectively determine near-optimal threshold values. A major

[1]Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad, Jevrejska 24, Belgrade 11000, Serbia. [2]Faculty of Science and Mathematics, University of Niš, Višegradska 33, Niš 18000, Serbia. ✉email: ivona.brajevic@mef.edu.rs

drawback of these algorithms is their reliance on algorithm-specific parameters. An increase in the number of parameters intensifies the complexity of parameter tuning. Each metaheuristic has its own shortcomings[12,13]. Some have a strong ability to explore novel areas of the search space, while others are better at exploiting previously discovered promising points. Various notable metaheuristic algorithms include particle swarm optimization (PSO), differential evolution (DE), Rao algorithms[14], artificial bee colony (ABC)[15], among others[12]. Additionally, numerous variants of these techniques have been developed to enhance their performance for specific types of problems[16,17].

In[18], the classic ABC and PSO metaheuristic algorithms were employed to optimize between-class variance and Kapur's entropy, aiming to address the multilevel thresholding problem. The performance of the ABC algorithm was compared with the basic PSO algorithm in terms of SSIM, PSNR, fitness function and computational time. A hybrid method that integrates the sine cosine algorithm (SCA) and artificial bee colony to search for thresholds using Otsu's function as the objective function is proposed in[19]. The effectiveness of the developed approach was evaluated against the SCA and ABC algorithms by measuring fitness values, computational time, SSIM and PSNR. An improved Bloch quantum artificial bee colony algorithm is developed to solve the problem of gray image multilevel threshold segmentation, as discussed in[20]. In that research, Kapur's entropy was employed as the objective function. The experimental results of the threshold segmentation achieved by the developed method were compared with those of the classic GA, PSO and ABC algorithms.

Each of the previously discussed studies focused on segmenting gray images using multilevel thresholding. On the other hand, color images provide a greater amount of information compared to grayscale images. Metaheuristic optimization methods are additionally extensively used for multilevel segmentation of color images. A hybrid method that incorporates Krill Herd algorithm into the ABC for multilevel color image segmentation is proposed in[21]. In that research, a modified objective function which combines Kapur's entropy with structural similarity index matrix is employed. The performance of the proposed method was assessed by comparing it to the ABC and PSO algorithms. This evaluation involved analysing boundary displacement error, peak signal-to-noise ratio and feature similarity index measurement. The implementation and comparison of the PSO, ABC, genetic algorithm, cuckoo search and modified whale optimization metaheuristics for multilevel segmentation of color images using Otsu's and Kapur's objective functions were presented in[22]. In[8], color images of plant diseases were segmented using the ABC, teaching-learning-based optimization, cuckoo search, teaching-learning-based artificial bee colony and its improved variant. The performance of the five metaheuristics algorithms was assessed and compared based on objective function values and three image quality metrics.

While the No Free Lunch Theorem indicates that no single algorithm is universally optimal[23], the ABC algorithm's adaptability, effective search strategies and minimal number of control parameters make it highly capable of tackling high-dimensional, nonlinear problems, such as multilevel thresholding[24,25]. Its straightforward design has inspired the development of numerous enhanced variants that retain its core simplicity while improving performance.

In this study, an enhanced ABC variant called globally informed ABC (giABC) is proposed to solve the multilevel color image thresholding problem. The proposed approach employed two novel search strategies in the employed and onlooker stages. Both search mechanisms use the mean value of the current better solutions to guide the search process. The search equation used in the onlooker phase additionally incorporates information from the global best solution to further enhance the exploitation capabilities of the standard ABC algorithm. Together, these enhancements enable giABC to achieve a robust balance between exploration and exploitation, yielding a significant improvement in the ABC algorithm. On the other hand, the proposed algorithm retains the structure and simplicity of the ABC, while not adding any additional control parameters compared to its original version adapted for solving integer programming problems.

The performance of the proposed approach is evaluated using twelve color images from Berkeley dataset. For comparison, the standard ABC algorithm, its two variants, the gbest-guided artificial bee colony (GABC)[26] and shuffle-based artificial bee colony (SB-ABC)[27], and the chaotic enhanced Rao (CER) algorithm[28] are also adjusted to solve the multilevel color image thresholding problem. The GABC is a well-established ABC variant[26], while the SB-ABC, originally designed for integer programming problems[27], effectively addresses discrete optimization challenges. The CER algorithm, which recently demonstrated superior performance in solving the multilevel thresholding problem[28], is a suitable choice for comparison.

The key contributions and advantages of this study can be outlined as follows:

- This study introduces an improved ABC variant, giABC, for solving the multilevel color thresholding problem using Otsu's function as the objective function.
- The performance of the giABC, standard ABC, its two variants and the CER algorithm is evaluated on twelve images from the Berkeley dataset.
- Experiments are conducted to segment benchmark images into 6, 8, 10, and 12 color threshold levels.
- The segmentation quality of the giABC, ABC, its two variants and the CER algorithm is assessed using metrics such as PSNR, SSIM, objective function value, computational time and statistical analysis.
- The diversity behavior of the ABC, GABC, SB-ABC and giABC in solving the multilevel color thresholding problem is analyzed.

The paper is organized as follows. The next section presents the formulation of the multilevel thresholding problem. Section "Overview of the algorithms used in the study" provides an overview of the metaheuristics employed for multilevel color image thresholding, including the proposed giABC algorithm. Section "Experimental study" presents and analyzes the experimental results. Section "Diversity behaviour analysis of ABC variants" provides a detailed analysis of the diversity behaviour of the ABC variants. The concluding section summarizes the key findings and insights of the study.

Formulation of multilevel thresholding problem

The multilevel thresholding problem aims to find the optimal k thresholds that divide the original image into $k + 1$ distinct regions, denoted as $R_0, R_1, \ldots, R_k$[8]. Let L represent the number of gray levels in a grayscale image or the red, green, and blue channels of an RGB image. Suppose that the gray levels of an image I are in the range $0, 1, \ldots, L-1$. Multilevel thresholding is characterized by:

$$\begin{aligned} R_0 &= \{(x,y) \in I \mid 0 \le f(x,y) \le t_1 - 1\}, \\ R_1 &= \{(x,y) \in I \mid t_1 \le f(x,y) \le t_2 - 1\}, \\ R_2 &= \{(x,y) \in I \mid t_2 \le f(x,y) \le t_3 - 1\}, \\ &\vdots \\ R_k &= \{(x,y) \in I \mid t_k \le f(x,y) \le L - 1\}. \end{aligned} \tag{1}$$

where k is the number of thresholds, $f(x, y)$ represents the gray level of the pixel (x, y) and t_i $(i = 1, \ldots, k)$ denotes the ith threshold value. Eq. (1) is commonly applied to grayscale images. Since each red, green and blue channel can be treated as an individual image, Eq. (1) can also be applied to the red, green and blue channels of RGB color images.

In the multilevel thresholding problem, the goal is to identify the optimal vector that optimizes a given objective function. For this purpose, Otsu's function is used as the objective function. The Otsu method is a technique that focuses on between-class variance and determines optimal thresholds by maximizing the variance among the segmented regions[29].

The probability distribution of the intensity values can be determined by analyzing the basic principles of the image's histogram[30]. Assume an image consists of N pixels, L denotes the number of gray levels in a grayscale image or the channel index within the RGB image, and h_i denotes the number of pixels that correspond to i-th intensity level, where i ranges from 0 to $L - 1$.

Then the probability distribution of the intensity values can be obtained by[30]:

$$P_i^c = \frac{{h_i}^c}{N} \tag{2}$$

where $\sum_{i=0}^{N} P_i^c = 1$. In the case of a grayscale image, $c = 1$, while for a color image, $c = 1, 2, 3$.

The aim of Otsu method is to maximize the following function[30]:

$$\begin{aligned} f(t_1, t_2, \ldots, t_k) &= \sigma_0^c + \sigma_1^c + \cdots + \sigma_k^c, \\ &\text{where } 0 \le t_i \le L - 1, \quad i = 1, 2, \ldots, k \end{aligned} \tag{3}$$

$$\begin{aligned} \sigma_0^c &= w_0^c({\mu_0}^c - {\mu_t}^c)^2, & {\mu_0}^c &= \sum_{i=0}^{t_1-1} \frac{i{P_i}^c}{w_0^c}, \\ \sigma_1^c &= w_1^c({\mu_1}^c - {\mu_t}^c)^2, & {\mu_1}^c &= \sum_{i=0}^{t_2-1} \frac{i{P_i}^c}{w_1^c}, \\ \sigma_2^c &= w_2^c({\mu_2}^c - {\mu_t}^c)^2, & {\mu_2}^c &= \sum_{i=t_2}^{t_3-1} \frac{i{P_i}^c}{w_2^c}, \\ &\vdots \\ \sigma_k^c &= w_k^c({\mu_k}^c - {\mu_t}^c)^2, & {\mu_k}^c &= \sum_{i=t_k}^{L-1} \frac{i{P_i}^c}{w_k^c}. \end{aligned} \tag{4}$$

where ${\mu_t}^c = \sum_{i=0}^{L-1} i{P_i}^c$ is the total mean intensity of the original image. For the multilevel color image thresholding using the Otsu's method, parameter $c = 1, 2, 3$ represents the image channels (Red, Green, Blue - R, G, B).

Overview of the algorithms used in the study

This section presents the ABC, GABC, SB-ABC and CER algorithms, along with our proposed giABC method, that were used to tackle the multilevel color image thresholding problem. These metaheuristics aim to find the optimal threshold values t_i $(i = 1, \ldots, k)$ which solve the problem formulated by Eq. (3).

The RGB color model is a straightforward and effective way to represent color images, relying on three basic color channels: red, green and blue[31]. Therefore, we need to determine the optimal thresholds and objective function values for each color component in the image. After the input image is provided, the histogram for each color channel (R, G, and B) is computed separately. Then, each tested method is independently executed on each channel to obtain the corresponding results.

Proposed ABC approach for multilevel thresholding

The traditional ABC algorithm is a population-based metaheuristic that, after the initialization of the population, executes three stages: employed, onlooker and scout, within a predefined number of iterations[18]. In addition to the common control parameters for all population-based metaheuristics, such as population size (*SP*) and maximum iteration number (*MIN*), the classic ABC features a specific control parameter *limit*. This parameter is used during the scout phase to regulate the exploration of new solutions.

The details of the ABC method applied to multilevel thresholding are outlined below.

Step 1. (Calculate the new population)

After initializing the control parameters, the ABC algorithm generates a population of *SP* solutions randomly across the search space. The lower and upper boundaries of the search space are defined as 0 and 255, respectively, corresponding to the intensity levels of the image. The set of solutions can be represented by matrix T.

$$T = [t_1, t_2, ..., t_{SP/2}], \qquad t_i = [t_{i,1}, t_{i,2}, ..., t_{i,k}] \tag{5}$$

where $i = \{1, 2, ..., SP/2\}$ is the index of a solution in the population and k is the number of thresholds to be determined for the multilevel thresholding problem. Each t_i represents a candidate solution, while $t_{i,j}$ denotes the *j*th threshold within solution t_i, for $j = \{1, 2, ..., k\}$. Each component *j*th is bounded value into [0, ..., 255] and the $t_{i,j} < t_{i,j+1}$ for all *j*. The objective function values for all solutions t_i are evaluated, current best solution is calculated and variable *iter* is set to 1.

Step 2. (Employed stage)

In the employed stage, every solution $t_i,\ i = 1, 2, ..., SP/2$, is subjected to an update procedure described as follows:

$$v_{i,j} = t_{i,j} + \varphi \cdot (t_{i,j} - t_{,}) \tag{6}$$

where *j* represents a randomly assigned parameter index, φ represents a uniformly distributed random number within the range (−1,1), t_l indicates the other component randomly chosen from the population. The boundary conditions of the potential solution v_i are checked after its creation. If a variable surpasses the defined search space limits, its value is corrected to the closest boundary within the allowed range. The update process finishes with a greedy choice between t_i and v_i.

Step 3. (Onlooker stage)

The individuals are chosen according to the probability specified by:

$$p_i = 0.9 \cdot (fit_i / maxfit) + 0.1 \tag{7}$$

where *maxfit* is the best fitness value of the population and fit_i denotes the fitness value of the *i*th solution in the population. A fitness value of the each solution $fit_i, i = 1, 2, ..., SP$, is calculated as follows:

$$fit_i = \begin{cases} 1/(1 + f(t_i)), & \text{if } f(t_i) \geq 0, \\ 1 + abs(f(t_i)) & , \ otherwise \end{cases} \tag{8}$$

where f(·) is the Otsu objective function (Eq. (3)) and abs(·) is the absolute value. During the onlooker phase, the individuals chosen for the update process are selected based on fitness proportionate selection. The update process in the onlooker stage is identical to that in the employed stage.

Step 4. (Scout stage)

Choose one of the least active solutions and replace it with a newly generated random solution. The ABC method computes $trial_i$ for each individual t_i over the search. These values indicate the number of failed attempts for each individual, which are used to decide on abandonment. In scout stage, if the highest *trial* value exceeds the *limit* parameter, the corresponding individual is renewed with a newly created random solution.

Step 5. (Store the best solution)

Store the best solution found so far (t_{best}) and increase the variable *iter* by one.

Step 6. (Verify the stopping condition)

If *iter* reaches the maximum number of iterations, terminate the algorithm; otherwise, return to Step 2.

Proposed GABC approach for multilevel thresholding

The GABC approach modifies the ABC search strategy by integrating information from the global best individual[26]. This approach uses the following modified ABC search equation in employed and onlooker stages:

$$v_{i,j} = t_{i,j} + \varphi_{i,j} \cdot (t_{i,j} - t_{h,j}) + \phi_{i,j} \cdot (y_j - t_{i,j}) \tag{9}$$

where v_i represent a new potential solution, t_i is current *i*th solution, t_h is another individual picked randomly from the population, y_j is the *j*th component of the global best individual, $\varphi_{i,j}$ is a uniformly distributed random number within the range (−1, 1) and $\phi_{i,j}$ is a uniformly distributed random number within the range [0, 1.5].

The GABC approach for multilevel image thresholding follows the same procedure outlined in the previous subsection. The sole difference is that in both bee phases, Eq. (6) is replaced by Eq. (9).

Proposed SB-ABC approach for multilevel thresholding

The SB-ABC algorithm utilizes modified ABC search operators in both the employed and onlooker stages[27]. In employed phase, the SB-ABC uses the following search strategy to produce a possible candidate individual v_i:

$$v_{i,j} = \begin{cases} t_{i,j} + \varphi_i \cdot (t_{i,j} - t_{h,j}) & , \text{ if } R_{i,j} < MR \\ t_{i,j} & , \text{ otherwise} \end{cases} \tag{10}$$

where $j = 1, 2, ..., k$. In Eq. (10) t_i is the current *i*th individual from the population and t_h denotes another solution chosen randomly. The *MR* parameter is a modification rate control parameter within the range (0, 1), where higher values increase the probability of changing more parameters in the parent solution. Also in this equation φ_i and R_{ij} are uniformly distributed random numbers, where φ_i is in (−1, 1) and R_{ij} is in (0,1).

In onlooker stage, the SB-ABC employs the following search equation to construct a possible new individual v_i:

$$v_{i,j} = \begin{cases} t_{i,j} + \varphi_{i,j} \cdot (t_{i,j} - t_{h,j}) \\ \quad + \phi_{i,j} \cdot (y_j - t_{i,j}), & \text{if } R_{i,j} < MR \\ t_{i,j}, & \text{otherwise} \end{cases} \tag{11}$$

where $j = 1, 2, ..., k$. In Eq. (11) t_i is the current *i*th individual, t_h indicates another solution selected at random from the population, y_j is the *j*th component of the current best individual and *MR* is modification rate parameter. In this equation φ_i and $R_{i,j}$ are random variables with a uniform distribution, where φ_i is in (−1, 1) and $R_{i,j}$ is in (0,1). In addition, in both bee stages, at every *RPPI*th cycle, the shuffle mutation operator is applied to create novel candidate solutions, where *RPPI* is a new control parameter called the random permutation production interval.

The SB-ABC approach consists of all the steps mentioned in the Sect. "Proposed ABC approach for multilevel thresholding". The only difference is that, in the employed phase, Eq. (6) is replaced by Eq. (10), while in the onlooker phase, Eq. (6) is replaced by Eq. (11).

Proposed CER approach for multilevel thresholding

The Rao-1 algorithm enhanced with chaotic behavior (CER) is proposed to solve multilevel thresholding problem[28]. The Chebyshev map has been used due to its superior performance compared to other chaotic maps. The CER is a metaphor-free optimization algorithm with two common control parameters, population size and maximum number of iterations.

The main steps of the CER algorithm involve initializing the population, identifying the best and worst solutions and updating the population of solutions. The current solution from the population t_i is updated using the following search equation:

$$v_{i,j} = t_{i,j} + z_l \cdot (best_j - worst_j) \tag{12}$$

where $j = 1, 2, ..., k$. In Eq. (12) *best* and *worst* represent the best and worst solutions in the population. Parameter z_l is the parameter obtained from the Chebyshev map by the next equation[28]:

$$z_{l+1} = \cos\left(l \cdot \cos^{-1}(z_l)\right) \tag{13}$$

During the population update process, the iteration loop starts, updating each solution from the population according to Eq. (12) and evaluating their objective function values. At the end of each iteration, the best and worst solutions are updated. The loop terminates when the maximum number of iterations is reached.

Proposed giABC approach for multilevel thresholding

In metaheuristics, exploitation refines existing solutions by focusing on promising regions of the search space, while exploration involves discovering new solutions farther from current ones. According to Eq. (6), the new solution is generated by moving the old one to a random location, promoting exploration but limiting exploitation.

To overcome these limitations, the proposed giABC algorithm introduces two modified search strategies in the employed and onlooker phases. Both search mechanisms incorporate a dynamically guided search vector, mt_{best}, while the onlooker phase further utilizes an additional exploitation term. Together, these enhancements enable giABC to achieve a robust balance between exploration and exploitation, leading to improved convergence speed and solution quality.

The proposed giABC algorithm, in the employed phase, utilizes the following modified search strategy to produce a candidate solution v_i:

$$v_{i,j} = \begin{cases} t_{i,j} + \varphi_i \cdot (mt_{best,j} - t_{h,j}) & , \text{ if } R_{i,j} < MR \\ t_{i,l} + \varphi_i \cdot (mt_{best,l} - t_{h,l}) & , \text{ otherwise} \end{cases} \tag{14}$$

where $j = 1, 2, ..., k$. In Eq. (14), t_i is the current *i*th individual, t_h is a randomly selected individual, is a randomly chosen index in $[1, k]$, φ_i is a random number in $[0, 1]$ fixed for all parameters of t_i and MR is the modification rate parameter.

The vector mt_{best}, which is used to guide the search process, is computed in each iteration as follows:

$$mt_{best} = \frac{y + \sum_{n \in S_i} t_n}{\text{length}(S_i) + 1} \tag{15}$$

where y is the current best individual, and S_i is the set of individuals with higher objective function values than t_i.

Updates based on mt_{best} focus on promising regions of the search space, while the random selection of maintains diversity, reducing the risk of premature convergence.

In the onlooker phase, the giABC algorithm uses the following modified mutation operator to create a candidate solution:

$$v_{i,j} = t_{i,j} + \varphi_{i,j} \cdot (mt_{best,j} - t_{h,j}) + \varphi_{i,j} \cdot (y_j - t_{i,j}) \tag{16}$$

where $j = 1, 2, ..., k$. In Eq. (16), $t_{i,j}$ is the jth dimension of the ith individual, t_h is a randomly selected individual, mt_{best} is the guiding vector calculated by Eq. (15), y is the current best individual in the population and $R_{i,j}$ is a random number with a uniform distribution in $[0, 1]$. The values of $\varphi_{i,j}$ are computed as the product of two uniformly distributed random numbers in $[0, 1]$, independently generated for each term in the equation.

The onlooker phase emphasizes exploitation by incorporating both mt_{best} and y in the search process. The additional term directs the algorithm more strongly toward the globally best solution, enhancing convergence speed. The values of $\varphi_{i,j}$ which favour smaller numbers, allow for precise adjustments during the search process. The introduced modifications make the onlooker phase more exploitation-focused compared to the employed phase.

The algorithm implements all three phases of the original ABC algorithm mentioned in the Sect. "Proposed ABC approach for multilevel thresholding". However, it modifies the employed phase by utilizing the search strategy described in Eq. (14), and the onlooker phase by employing the equation defined in Eq. (16).

The giABC algorithm retains the core structure of the original ABC algorithm, introducing improvements without increasing its complexity. It adds only one additional control parameter, MR, which is commonly used in applications of ABC for integer programming problems. This ensures that giABC maintains the simplicity of ABC while significantly enhancing its performance.

Experimental study

This section presents results obtained by the CER, ABC, GABC, SB-ABC and giABC metaheuristics used to solve the multilevel color image thresholding problem. Extensive experiments were conducted on twelve color test images from the Berkeley Segmentation Dataset[32] to evaluate the effectiveness of these algorithms for multilevel color image thresholding. The test images include: #147091 (tree), #157055 (couple), #182053 (train), #148089 (gate), #197017 (horses), #97033 (house), #260058 (pyramid), #253027 (zebras), #223061 (facade), #38082 (deer), #19021 (cactus), #86068 (fishes). For simplicity, these images were referred to by their respective names in parentheses throughout the paper. These images are segmented using threshold color values of 6, 8, 10 and 12.

Metrics for evaluating image quality

The Peak Signal-to-Noise Ratio (PSNR) is an important metric for evaluating image segmentation quality[33]. It measures the ratio between maximum signal strength and noise level, with the result expressed in decibels. A higher PSNR indicates better thresholding quality. The PSNR is calculated by[8]:

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}} \right) \tag{17}$$

The mean square error (MSE) is defined by the following formula:

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [I(i,j) - S(i,j)]^2 \tag{18}$$

where $m \times n$ denotes the image size, while $S(i, j)$ and $I(i, j)$ represent the segmented and actual images, respectively. For an RGB color image, the PSNR is calculated separately for each of the three primary color channels, and the overall PSNR for the color image is taken as the average of these individual values[31].

The Structural Similarity Index Method (SSIM) is a perception-based model that views image degradation as a change in perceived structural information[33]. The similarity between the original image and the segmented images can be described as follows:

$$SSIM = \frac{(2\mu_I \mu_S + C_1)(2\sigma_{IS} + C_2)}{(\mu_I^2 + \mu_S^2 + C_1)(\sigma_I^2 + \sigma_S^2 + C_2)} \tag{19}$$

where μ_I and μ_S are the averages of I and S, respectively, and σ_I and σ_S represent the variances of I and S[31]. The local correlation coefficient between I and S is denoted by σ_{IS}, while C_1 and C_2 are constants.

The SSIM can be adapted for RGB color images, as demonstrated below[31]:

$$SSIM = \sum_{c} SSIM(I^T, S^T) \tag{20}$$

where I and S represent the original image on the Tth channel and the multilevel thresholded image on the Tth channel, respectively, where T is the channel number. A higher SSIM value signifies improved quality in the thresholding process.

Experimental configuration

To ensure a fair comparison, all five algorithms, the CER, ABC, GABC, SB-ABC and giABC, are set to operate with a population size of 40 and a maximum of 4000 function evaluations. The initial population for each metaheuristic approach is generated randomly and uniformly over the interval [0,255].

In the ABC and GABC algorithms, the control parameter *limit* is set to 50, based on the original paper[18]. For the SB-ABC algorithm, the control parameters are configured with *limit* set to 50, *RPPI* to 3 and *MR* to 0.8, as recommended in the respective study[27]. In the giABC algorithm, the parameter *limit* is set to 50, as suggested in the original paper[18], while the parameter *MR* is set to 0.8, as recommended in the previous study[34].

Each of the five algorithms is repeated 40 times independently for each image, each channel and for each k value. The algorithms were implemented in Java programming language. The tested algorithms were executed on a PC equipped with an Intel(R) Core(TM) i5-4460 processor running at 3.2 GHz, 16 GB of RAM, and a Windows operating system. The performance evaluation metrics include the objective function value, peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) and computation time.

Objective function value comparison

To evaluate the robustness of the proposed algorithms, the mean objective function values were calculated. These values were obtained by averaging the threshold values for the three color channels (R, G, B) across 40 independent runs for each image.

The procedure was repeated for 12 images at threshold levels of 6, 8, 10 and 12, ensuring a comprehensive evaluation of algorithm performance under varying segmentation complexities. This resulted in a single representative mean value for each algorithm, capturing its overall stability and consistency across the dataset.

Table 1 presents a comparison of the mean Otsu threshold values across the color channels (R, G, B) calculated over 40 independent runs for the CER, ABC, GABC, SB-ABC and giABC algorithms.

As presented in Table 1, giABC consistently achieves the best average results across all thresholds ($k = 6, 8, 10, 12$) and channels (R, G, B), with a final rank of 1.0, demonstrating superior robustness. The GABC has a final rank of 2.0, demonstrating strong performance in terms of the stability of the obtained results. In contrast, SB-ABC, ABC and CER exhibit lower stability, with ranks of 3.0, 4.0 and 5.0, reflecting their limitations in maintaining consistent performance compared to the GABC and giABC variants.

Figure 1 shows the convergence behavior of the tested algorithms across different color channels (R, G, B) for two selected images. The proposed giABC algorithm demonstrates superior performance, achieving faster convergence and higher objective values compared to other algorithms, particularly CER. While the giABC and SB-ABC show competitive early performance, giABC achieves the most stable and accurate results, ensuring robustness across all color channels.

PSNR and SSIM comparison

The PSNR and SSIM values for the twelve test images were assessed following segmentation with thresholds that maximize Otsu's objective function. Table 2 displays the PSNR values and performance rankings for the CER, ABC, GABC, SB-ABC and giABC approaches. As mentioned earlier, a high PSNR value signifies high quality in the segmented image, which in turn indicates strong performance of the optimization method.

As indicated in Table 2, the giABC algorithm demonstrated the highest performance with respect to PSNR, achieving a final rank of 1.9. The SB-ABC and GABC algorithms showed competitive results, with final ranks of 2.5 and 2.9, respectively, all outperforming the ABC algorithm, which achieved a rank of 3.7. The CER method, with a final rank of 4.1, showed relatively lower PSNR values compared to the other algorithms.

Additionally, a high SSIM value reflects strong performance of the optimization method. Based on the SSIM values presented in Table 3, the algorithms can be ranked from best to worst as giABC, SB-ABC, GABC, ABC and CER, with final rankings of 2.0, 2.3, 2.8, 3.6 and 4.2, respectively.

To examine the differences between the proposed giABC algorithm and other algorithms (SB-ABC, GABC, ABC and CER) for the mean, PSNR and SSIM metrics, we performed the Wilcoxon signed-rank test at a significance level of 0.05[35]. All p values were calculated using the R software package (version 4.1.2).

The results of the Wilcoxon signed-rank test are presented in Table 4, which includes the names of the compared approaches, the p values, and the decision on the null hypothesis for the mean, PSNR and SSIM metrics. The symbol '+' denotes that the first algorithm is significantly better than the second, '–' indicates that the first algorithm is significantly worse, and '$\approx$' signifies that there is no significant difference between the two algorithms.

As shown in Table 4, the p values indicate that the proposed giABC algorithm performs significantly better than each of the other tested algorithms (SB-ABC, GABC, ABC and CER) for mean, PSNR and SSIM metrics. These results highlight the ability of the proposed giABC algorithm to achieve superior performance in segmentation quality compared to the other approaches.

Computational time comparison

In this subsection, the computational times of the CER, ABC, GABC, SB-ABC and giABC algorithms have been examined. The average computational time for these algorithms over 40 runs at level 10 is provided in Table 5 as an example.

As shown in Table 5, the five approaches demonstrate execution times ranging from 60 to 175 ms, indicating that all algorithms exhibit computational times suitable for most applications. The CER algorithm demonstrated

Image	k	CER	ABC	GABC	SB-ABC	giABC
Tree	6	4859.9033	4876.5549	4876.7651	4876.6100	4876.7962
	8	4890.3530	4905.2001	4905.7636	4905.5464	4905.8932
	10	4906.8139	4920.3463	4921.1796	4920.7512	4921.3329
	12	4917.5321	4928.8011	4929.4544	4929.1108	4929.8417
Couple	6	3601.8945	3612.5212	3612.7983	3612.7695	3612.8471
	8	3624.7707	3641.7794	3642.2945	3642.1923	3642.3787
	10	3640.8854	3655.8430	3656.8610	3656.6203	3657.0518
	12	3651.9828	3664.2129	3665.1776	3664.8916	3665.4214
Train	6	2938.2848	2955.4035	2955.5050	2955.4883	2955.5086
	8	2967.0765	2980.2403	2980.7058	2980.4001	2980.7017
	10	2981.0184	2993.6988	2994.5071	2994.0599	2994.6735
	12	2991.1845	3001.5436	3002.0408	3001.7149	3002.3660
Gate	6	3660.6080	3677.0130	3677.3040	3677.2176	3677.3570
	8	3690.8207	3707.3913	3708.1509	3707.6800	3708.3024
	10	3708.7962	3722.5214	3723.3458	3722.8053	3723.7608
	12	3718.4767	3731.3513	3731.9225	3731.5036	3732.2350
Horses	6	4211.6579	4225.7851	4225.9312	4225.8100	4225.9521
	8	4236.1157	4250.8090	4251.1841	4251.0080	4251.2317
	10	4250.8080	4264.0036	4264.7005	4264.2547	4264.8041
	12	4260.3979	4271.8965	4272.6558	4272.3119	4272.9420
House	6	5865.2798	5885.3082	5885.3781	5885.2361	5885.3788
	8	5904.7167	5915.3823	5915.8119	5915.6391	5915.9051
	10	5922.8398	5933.3558	5934.0493	5933.9525	5934.1800
	12	5927.5748	5943.7444	5944.4150	5944.4111	5944.6401
Pyramid	6	1161.1855	1172.8416	1173.0146	1172.8488	1173.0152
	8	1175.6041	1187.6567	1188.1497	1187.7445	1188.4015
	10	1185.5482	1194.9909	1195.8182	1195.1012	1196.0894
	12	1191.5761	1199.4767	1200.4433	1199.9097	1200.8832
Zebras	6	1650.8544	1664.7541	1664.9082	1664.8566	1664.9175
	8	1672.1639	1684.7165	1685.3398	1685.0926	1685.4100
	10	1683.4785	1694.6483	1695.3812	1695.1431	1695.5604
	12	1691.0116	1700.3457	1701.2331	1700.9812	1701.4609
Facade	6	3726.0243	3744.2400	3744.4537	3744.3795	3744.4617
	8	3755.5129	3773.4348	3773.9274	3773.7643	3773.9944
	10	3772.2689	3787.9347	3788.7209	3788.2796	3788.8832
	12	3783.2284	3796.2698	3797.1557	3796.7849	3797.4102
Deer	6	1145.2634	1159.5526	1159.7663	1159.4298	1159.8016
	8	1161.8623	1175.6514	1176.1163	1174.9924	1176.3960
	10	1171.8749	1183.4943	1184.3292	1182.7806	1184.9528
	12	1178.2977	1188.1040	1188.7987	1187.8724	1189.6049
Cactus	6	2155.6813	2169.8072	2169.9772	2169.9165	2169.9949
	8	2179.2314	2193.7225	2194.1539	2193.9837	2194.2412
	10	2188.0704	2205.2288	2205.9348	2205.7119	2206.0963
	12	2201.4624	2211.5532	2212.4349	2212.1720	2212.7157
Fishes	6	1035.3956	1055.7673	1055.8909	1055.8617	1055.9043
	8	1061.0980	1071.0967	1071.7036	1071.4334	1071.7710
	10	1069.3212	1078.6203	1079.6517	1079.1364	1079.7747
	12	1075.1456	1082.9400	1084.1164	1083.6488	1084.4607
Avg. rank per threshold	6	5.0	4.0	2.0	3.0	1.0
	8	5.0	4.0	2.0	3.0	1.0
	10	5.0	4.0	2.0	3.0	1.0
	12	5.0	4.0	2.0	3.0	1.0
Final rank		5.0	4.0	2.0	3.0	1.0

Table 1. Comparison of the mean Otsu threshold values across the color channels (R, G, B) for each algorithm (CER, ABC, GABC, SB-ABC and giABC), computed over 40 independent runs.

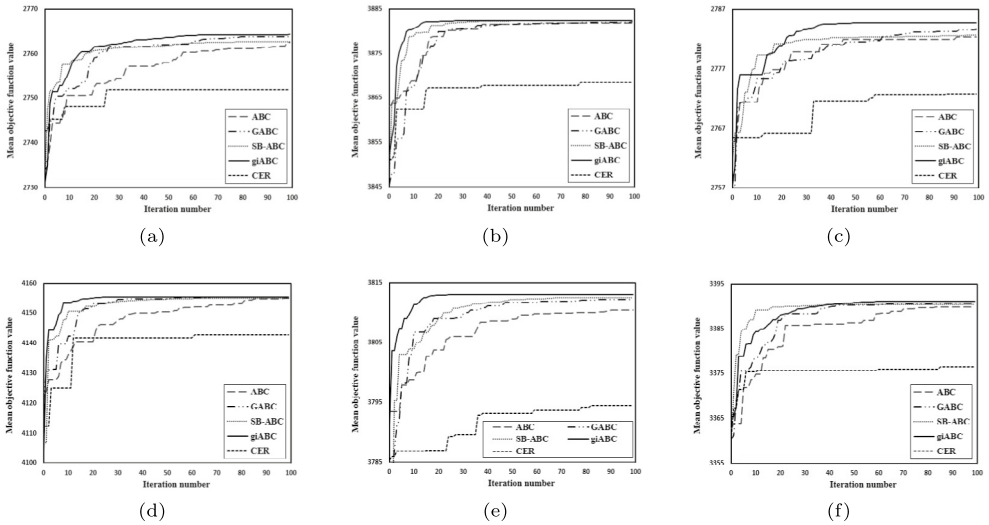

Fig. 1. Convergence curves for the ABC, GABC, SB-ABC, giABC and CER on certain images across different channels and threshold levels: (**a**) Tree (red, k = 8), (**b**) Tree (red, k = 10), (**c**) Tree (red, k = 12), (**d**) Facade (red, k = 8), (**e**) Facade (green, k = 10), (**f**) Facade (blue, k = 12).

the fastest computational time but at the cost of lower segmentation quality. In contrast, SB-ABC required the most time, while the computational times of ABC, GABC, and giABC were comparable, with giABC being slightly slower but offering superior segmentation quality.

To further enhance the efficiency of metaheuristic optimization algorithms, approaches such as optimizing stopping criteria[36] and leveraging parallel computing techniques on graphics processing units[37] can be explored. These strategies could reduce computational time while maintaining the quality of results.

The three ABC variants consistently outperformed the standard ABC and CER algorithms in terms of accuracy, PSNR, SSIM, and stability. Among them, giABC demonstrated the most accurate and stable segmentation results, achieving the highest PSNR and SSIM values. This makes giABC particularly suitable for complex multilevel thresholding tasks in fields such as medical imaging, remote sensing, and object recognition.

Diversity behaviour analysis of ABC variants

This section analyzes the diversity performance of the ABC methods for color multilevel thresholding.

Diversity points to divergences between agents of population[38]. Varied population is essential for exploring novel areas of a search space. However, encouraging diversity in each generation may lead to improper ratio between exploiting previously found promising points and exploration.

The following diversity metric is employed to evaluate the differences among agents in the whole population[39]:

$$Div_j = \frac{1}{SP}\sum_{i=1}^{SP} med_j - x_{i,j} \tag{21}$$

$$Div(t) = \frac{1}{D}\sum_{i=1}^{D} Div_j \tag{22}$$

In the Eq. (21), med_j represents the median of *j*th component across the entire population, $x_{i,j}$ denotes *j*th component of *i*th solution individual and *SP* is the total number of individuals. Value Div_j indicates diversity in the *j*th dimension. The Eq. (22) is used to calculate the population diversity in *t*th iteration.

Diversity performance of the ABC, GABC, SB-ABC and giABC, for two test images with 10 thresholds for the red, green and blue channels is demonstrated in Fig. 2. As shown in the Fig. 2, all algorithms exhibit a decline in diversity during the search process. This behaviour is expected, as the algorithms progressively converge toward optimal solutions, reducing the diversity within the population.

Figure 2 illustrates that the diversity in the ABC algorithm remains higher than that of the GABC, SB-ABC and giABC variants at each iteration of the search process. This higher diversity in the ABC method is attributed to its limited exploitation ability, as its solution search strategy depends on a randomly selected neighbouring food source. In contrast, the three analysed ABC variants utilize the best solution found so far to guide the subsequent search process.

Specifically, the differences among agents in SB-ABC are lower in comparison with other variants of ABC. This indicates that favorable regions of the search space in SB-ABC are often identified in the early stages of the search. As a result, this algorithm may, in some runs, get trapped in a local minimum, while in others, it has the potential to find highly accurate solutions. In later iterations, the diversity in giABC is lower than in GABC but

Image	k	CER	ABC	GABC	SB-ABC	giABC
Tree	6	30.4364	30.7576	30.7713	30.7713	30.7713
	8	32.5999	32.8832	32.8725	32.9856	32.9856
	10	34.0342	34.8975	34.8975	34.9639	34.9639
	12	35.4670	36.4951	36.4951	36.5222	36.5000
Couple	6	28.8540	28.9712	29.1546	29.1546	29.1546
	8	30.3033	30.6918	30.7312	30.7312	30.7312
	10	31.2976	31.6884	32.0842	32.1137	32.1137
	12	31.8315	33.0792	33.1303	33.1616	33.1435
Train	6	30.8504	30.5983	30.6837	30.6837	30.6837
	8	32.8119	33.5016	33.5269	33.5269	33.5269
	10	34.6977	35.3074	35.2972	35.4134	35.4014
	12	35.9397	37.1520	37.1402	37.2837	37.3398
Gate	6	26.3436	30.0341	30.4262	30.4240	30.4240
	8	33.7637	34.2290	34.3347	34.3347	34.3347
	10	35.9373	36.4920	36.4903	36.6561	36.6666
	12	36.6659	35.2062	35.2569	35.2225	35.3150
Horses	6	28.0653	28.5530	28.5858	28.5858	28.5858
	8	30.6902	30.6816	30.6804	30.6908	30.6908
	10	30.9723	33.1310	33.2320	33.2642	33.2654
	12	34.0326	34.8266	34.8713	34.8141	34.8865
House	6	30.0540	29.9582	29.9582	29.9582	29.9582
	8	32.0458	32.6963	32.7660	32.7660	32.7660
	10	34.1216	35.1054	35.1713	35.2748	35.2748
	12	35.3372	36.4536	36.9461	36.9198	36.9004
Pyramid	6	28.3815	28.2342	28.3079	28.3079	28.3079
	8	32.0031	31.2154	30.5806	31.2762	31.2762
	10	32.9597	33.6324	33.6495	33.9052	33.9732
	12	34.3675	36.6156	36.8728	36.5534	37.0079
Zebras	6	28.8949	27.9026	28.0200	28.0200	28.0200
	8	29.8262	30.4789	30.4869	30.4869	30.4869
	10	32.6115	32.5127	32.3718	32.5925	32.5944
	12	33.1114	33.3965	34.0889	34.0591	34.0591
Fascade	6	28.7297	27.9067	27.8977	27.8977	27.8977
	8	30.2063	30.1416	30.1007	30.2313	30.2313
	10	32.6366	31.4686	31.6778	31.6769	31.6827
	12	32.8409	33.5103	33.4914	33.4322	33.5200
Deer	6	29.6661	28.8816	28.8816	28.8816	28.8816
	8	29.6801	31.1279	31.1412	31.1363	31.1412
	10	33.2299	33.0922	33.2075	33.2076	33.4029
	12	32.6291	34.6989	34.7179	34.7430	34.9403
Cactus	6	30.6384	30.7134	30.7134	30.7134	30.7134
	8	32.0572	32.5290	32.5033	32.5033	32.5033
	10	33.6708	33.4472	33.3884	33.0830	33.4559
	12	33.6418	33.8738	33.5698	33.9330	33.9401
Fishes	6	29.1191	29.4941	29.4941	29.4941	29.4941
	8	30.5610	30.6523	30.7603	30.7603	30.7603
	10	32.6085	31.7218	31.7514	31.7135	31.7510
	12	32.4985	32.4484	32.5164	32.4983	32.3669
Avg. rank per threshold	6	3.9	3.6	2.5	2.5	2.5
	8	4.2	3.6	2.9	2.1	2.0
	10	3.6	3.9	3.5	2.6	1.6
	12	4.5	3.6	2.9	2.6	1.5
Final rank		4.1	3.7	2.9	2.5	1.9

Table 2. Comparison of PSNR calculated by CER, ABC, GABC, SB-ABC and giABC.

Image	k	CER	ABC	GABC	SB-ABC	giABC
Tree	6	0.8778	0.8800	0.8800	0.8800	0.8800
	8	0.8970	0.9065	0.9051	0.9056	0.9056
	10	0.9191	0.9252	0.9255	0.9260	0.9260
	12	0.9312	0.9403	0.9408	0.9408	0.9412
Couple	6	0.9074	0.9082	0.9105	0.9105	0.9105
	8	0.9319	0.9347	0.9361	0.9361	0.9361
	10	0.9449	0.9462	0.9476	0.9476	0.9476
	12	0.9494	0.9547	0.9551	0.9547	0.9549
Train	6	0.9392	0.9377	0.9387	0.9387	0.9387
	8	0.9428	0.9563	0.9564	0.9564	0.9564
	10	0.9551	0.9577	0.9576	0.9579	0.9578
	12	0.9634	0.9665	0.9679	0.9685	0.9690
Gate	6	0.8982	0.9436	0.9458	0.9462	0.9462
	8	0.9682	0.9677	0.9682	0.9682	0.9682
	10	0.9749	0.9794	0.9791	0.9794	0.9795
	12	0.9767	0.9698	0.9707	0.9707	0.9709
Horses	6	0.8701	0.8660	0.8669	0.8669	0.8669
	8	0.8936	0.8965	0.8976	0.8980	0.8980
	10	0.9165	0.9273	0.9288	0.9309	0.9309
	12	0.9427	0.9466	0.9473	0.9465	0.9478
House	6	0.9003	0.9009	0.9009	0.9009	0.9009
	8	0.9263	0.9360	0.9375	0.9375	0.9375
	10	0.9519	0.9541	0.9560	0.9565	0.9565
	12	0.9561	0.9658	0.9661	0.9656	0.9654
Pyramid	6	0.8695	0.8737	0.8743	0.8743	0.8743
	8	0.9175	0.9206	0.8509	0.9208	0.9208
	10	0.9363	0.9431	0.9452	0.9458	0.9463
	12	0.9475	0.9647	0.9654	0.9650	0.9658
Zebras	6	0.9027	0.9037	0.9038	0.9038	0.9038
	8	0.9307	0.9385	0.9385	0.9385	0.9385
	10	0.9507	0.9564	0.9569	0.9586	0.9586
	12	0.9645	0.9645	0.9688	0.9681	0.9681
Fascade	6	0.8916	0.8936	0.8919	0.8919	0.8919
	8	0.9137	0.9160	0.9154	0.9159	0.9159
	10	0.9353	0.9324	0.9341	0.9344	0.9344
	12	0.9458	0.9458	0.9466	0.9463	0.9475
Deer	6	0.9450	0.9358	0.9358	0.9358	0.9358
	8	0.9506	0.9610	0.9611	0.9611	0.9611
	10	0.9782	0.9734	0.9745	0.9743	0.9754
	12	0.9721	0.9806	0.9745	0.9815	0.9812
Cactus	6	0.9280	0.9317	0.9317	0.9317	0.9317
	8	0.9464	0.9509	0.9505	0.9505	0.9505
	10	0.9563	0.9596	0.9599	0.9571	0.9605
	12	0.9596	0.9630	0.9615	0.9633	0.9633
Fishes	6	0.9282	0.9351	0.9351	0.9351	0.9351
	8	0.9489	0.9533	0.9536	0.9536	0.9536
	10	0.9595	0.9619	0.9624	0.9627	0.9626
	12	0.9618	0.9665	0.9662	0.9666	0.9667
Avg. rank per threshold	6	3.7	3.5	2.7	2.6	2.6
	8	4.6	3.5	2.8	2.1	2.1
	10	4.0	4.0	3.2	2.2	1.7
	12	4.6	3.7	2.5	2.5	1.8
Final rank		4.2	3.6	2.8	2.3	2.0

Table 3. Comparison of SSIM calculated by CER, ABC, GABC, SB-ABC and giABC.

Algorithm	Mean		PSNR		SSIM	
	p value	Dec.	*p* value	Dec.	*p* value	Dec.
giABC versus SB-ABC	7.00E−15	+	3.00E−02	+	1.00E−02	+
giABC versus GABC	4.00E−12	+	2.00E−04	+	2.00E−04	+
giABC versus ABC	7.00E−15	+	8.00E−08	+	3.00E−06	+
giABC versus CER	7.00E−15	+	6.00E−04	+	9.00E−07	+

Table 4. Wilcoxon's rank-sum test results for the mean value, PSNR and SSIM metrics comparing giABC with other algorithms.

Image	CER	ABC	GABC	SB-ABC	giABC
Tree	66.4	82.0	94.3	160.1	96.4
Couple	72.8	90.0	89.1	147.2	102.7
Train	78.4	98.0	101.6	174.0	103.3
Gate	64.1	79.1	79.4	141.7	89.4
Horses	72.2	89.1	89.9	156.7	99.4
House	68.7	84.9	93.9	124.3	97.1
Pyramid	64.2	80.3	82.4	137.1	95.5
Zebras	71.4	88.8	92.8	138.0	99.1
Facade	69.2	85.2	87.4	118.2	96.2
Deer	66.3	82.5	86.1	117.6	95.1
Cactus	60.1	74.1	74.8	105.1	84.3
Fishes	61.3	76.4	77.4	111.2	86.5

Table 5. Average computational times in milliseconds for the CER, ABC, GABC, SB-ABC, and giABC algorithms at level 10.

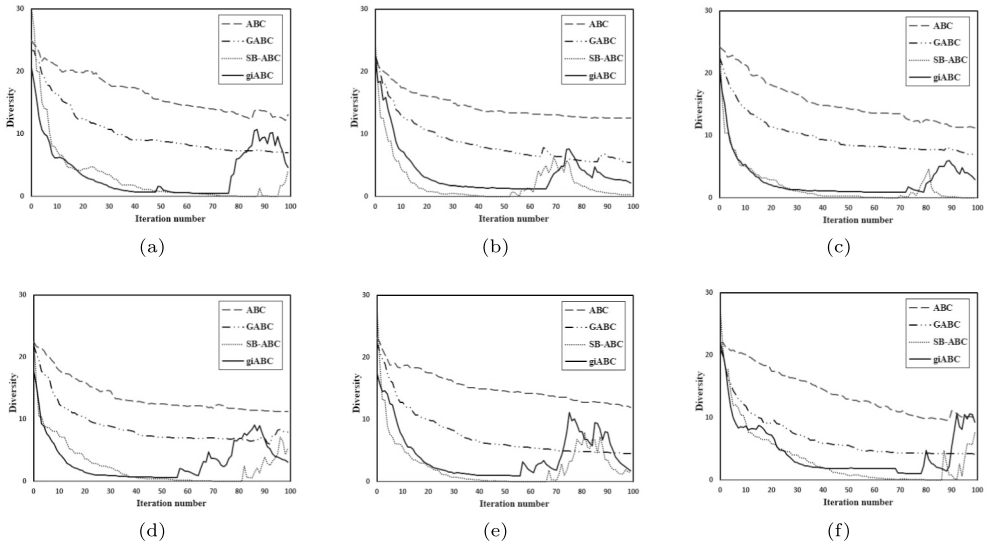

Fig. 2. Diversity performance of the ABC, GABC, SB-ABC and giABC for certain images with 10 thresholds: (**a**) Tree (red), (**b**) Tree (green), (**c**) Tree (blue), (**d**) Gate (red), (**e**) Gate (green), (**f**) Gate (blue).

remains higher than in SB-ABC, as shown in Fig. 2. This balance in diversity allows giABC to avoid premature convergence better than SB-ABC, while still maintaining a competitive level of exploration.

Overall, the results demonstrate that giABC achieves a superior balance between maintaining sufficient diversity and driving convergence. This makes it less prone to premature convergence while still being capable of achieving highly competitive solutions across all tested scenarios.

Conclusion

This paper presents the globally informed artificial bee colony (giABC), an enhanced ABC variant proposed for multilevel color image thresholding. By incorporating two novel mutation operators, giABC introduces dynamic guidance toward the mean of better solutions in the employed phase and combines global best attraction with adaptation to promising solutions in the onlooker phase. These modifications ensure a sustained balance between exploration and exploitation, resulting in increased convergence speed and improved solution quality. The proposed giABC, along with the standard ABC, its two variants, and the chaotically-enhanced Rao algorithm, were evaluated on twelve benchmark color images using Otsu's objective function.

Comprehensive evaluations based on the objective function, PSNR and SSIM showed that giABC consistently outperformed competing methods across all metrics. Furthermore, the giABC algorithm proved particularly effective for complex multilevel thresholding tasks at higher threshold values, with statistical validation confirming its significance. An analysis of computational time revealed that the CER algorithm is the most efficient, although all algorithms demonstrated processing times suitable for most applications.

In future work, further improvements in computational efficiency can be achieved by optimizing stopping criteria to minimize unnecessary iterations and exploring parallel computing techniques to accelerate the performance of the algorithms. Expanding the image dataset to include medical and satellite images, as well as applying alternative thresholding criteria could be considered to further validate the effectiveness of the proposed giABC algorithm. Additionally, future research could investigate developing hybrid approaches that combine giABC with other deterministic or stochastic methods to address other complex optimization problems.

Data availability

The datasets used in the current study are publicly available at https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html.

Code availability

The source code used in this work is available from the corresponding author on request.

Materials availability

Not applicable.

Received: 5 February 2025; Accepted: 2 June 2025
Published online: 01 July 2025

References

1. Wang, D. & Wang, X. P. The iterative convolution-thresholding method (ICTM) for image segmentation. *Pattern Recognit.* **130**, 108794. https://doi.org/10.1016/j.patcog.2022.108794 (2022).
2. Ramesh, K. K. D., Kumar, G. K., Swapna, K., Datta, D. & Rajest, S. S. A review of medical image segmentation algorithms. *EAI Endorsed Trans. Perv. Health Tech.* **7**, 6. https://doi.org/10.4108/eai.12-4-2021.169184 (2021).
3. Brar, K. K. et al. Image segmentation review: Theoretical background and recent advances. *Inf. Fusion* **114**, 102608. https://doi.org/10.1016/j.inffus.2024.102608 (2025).
4. Jiang, Y., Zhang, D., Zhu, W. & Wang, L. Multi-level thresholding image segmentation based on improved slime Mould algorithm and symmetric cross-entropy. *Entropy* **25**, 178. https://doi.org/10.3390/e25010178 (2023).
5. Ishak, A. B. Choosing parameters for Rényi and Tsallis entropies within a two-dimensional multilevel image segmentation framework. *Phys. A: Stat. Mech. Appl.* **466**, 521–536. https://doi.org/10.1016/j.physa.2016.09.053 (2017).
6. Merzban, M. H. & Elbayoumi, M. Efficient solution of Otsu multilevel image thresholding: A comparative study. *Expert Syst. Appl.* **116**, 299–309. https://doi.org/10.1016/j.eswa.2018.09.008 (2019).
7. Houssein, E. H., Mohamed, G. M., Ibrahim, I. A. & Wazery, Y. M. An efficient multilevel image thresholding method based on improved heap-based optimizer. *Sci. Rep.* **13**, 9094. https://doi.org/10.1038/s41598-023-36066-8 (2023).
8. Akay, R., Saleh, R. A. A., Farea, S. M. O. & Kanaan, M. Multilevel thresholding segmentation of color plant disease images using metaheuristic optimization algorithms. *Neural Comput. Appl.* **34**, 1161–1179. https://doi.org/10.1007/s00521-021-06437-1 (2022).
9. Rai, R., Das, A. & Dhal, K. G. Nature-inspired optimization algorithms and their significance in multi-thresholding image segmentation: an inclusive review. *Evol. Syst.* **13**, 889–945. https://doi.org/10.1007/s12530-022-09425-5 (2022).
10. Mousavirad, S. J., Schaefer, G., Zhou, H. & Moghadam, M. H. How effective are current population-based metaheuristic algorithms for variance-based multi-level image thresholding?. *Knowl. Based Syst.* **272**, 110587. https://doi.org/10.1016/j.knosys.2023.110587 (2023).
11. Hu, P., Han, Y., Zhang, Z., Chu, S. C. & Pan, J. S. A multi-level thresholding image segmentation algorithm based on equilibrium optimizer. *Sci. Rep.* **14**, 29728. https://doi.org/10.1038/s41598-024-81075-w (2024).
12. Halim, A. H., Ismail, I. & Das, S. Performance assessment of the metaheuristic optimization algorithms: An exhaustive review. *Artif. Intell. Rev.* **54**, 2323–2409. https://doi.org/10.1007/s10462-020-09906-6 (2021).
13. Thakur, G. & Pal, A. A novel slime mould multiverse algorithm for global optimization and mechanical engineering design problems. *Int. J. Comput. Intell. Syst.* **17**, 308. https://doi.org/10.1007/s44196-024-00704-4 (2024).
14. Rao, R. Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *Int. J. Ind. Eng. Comput.* **11**, 107–130. https://doi.org/10.5267/j.ijiec.2019.6.002 (2020).
15. Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **39**, 459–471. https://doi.org/10.1007/s10898-007-9149-x (2007).
16. Brajević, I. & Ignjatović, J. An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems. *J. Intell. Manuf.* **30**, 2545–2574. https://doi.org/10.1007/s10845-018-1419-6 (2019).
17. Tian, Y. et al. Application of hybrid algorithm based on ant colony optimization and sparrow search in UAV path planning. *Int. J. Comput. Intell. Syst.* **17**, 286. https://doi.org/10.1007/s44196-024-00652-z (2024).
18. Akay, B. A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Appl. Soft Comput.* **13**, 3066–3091. https://doi.org/10.1016/j.asoc.2012.03.072 (2013).
19. Ewees, A. A., Elaziz, M. A., Al-Qaness, M. A. A., Khalil, H. A. & Kim, S. Improved artificial bee colony using sine-cosine algorithm for multi-level thresholding image segmentation. *IEEE Access* **8**, 26304–26315. https://doi.org/10.1109/ACCESS.2020.2971249 (2020).

20. Huo, F., Sun, X. & Ren, W. Multilevel image threshold segmentation using an improved bloch quantum artificial bee colony algorithm. *Multimed. Tools Appl.* **79**, 2447–2471. https://doi.org/10.1007/s11042-019-08231-7 (2020).
21. Zhang, S., Jiang, W. & Satoh, S. Multilevel thresholding color image segmentation using a modified artificial bee colony algorithm. *IEICE Trans. Inf. Syst.* **E101.D**, 2064–2071. https://doi.org/10.1587/transinf.2017EDP7183 (2018).
22. Anitha, J., Pandian, S. I. A. & Agnes, S. A. An efficient multilevel color image thresholding based on modified whale optimization algorithm. *Expert Syst. Appl.* **178**, 115003. https://doi.org/10.1016/j.eswa.2021.115003 (2021).
23. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**, 67–82. https://doi.org/10.1109/4235.585893 (1997).
24. Kaya, E., Gorkemli, B., Akay, B. & Karaboga, D. A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. *Eng. Appl. Artif. Intell.* **115**, 105311. https://doi.org/10.1016/j.engappai.2022.105311 (2022).
25. Devadason, J. R., Hepsiba, P. S. & Solomon, D. G. Case studies on the applications of the artificial bee colony algorithm. *Sādhanā* **49**, 152. https://doi.org/10.1007/s12046-024-02498-9 (2024).
26. Zhu, G. & Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **217**, 3166–3173. https://doi.org/10.1016/j.amc.2010.08.049 (2010).
27. Brajević, I. A shuffle-based artificial bee colony algorithm for solving integer programming and minimax problems. *Mathematics* **9**, 1211. https://doi.org/10.3390/math9111211 (2021).
28. Olmez, Y., Sengur, A., Koca, G. O. & Rao, R. V. An adaptive multilevel thresholding method with chaotically-enhanced Rao algorithm. *Multimed. Tools Appl.* **82**, 12351–12377. https://doi.org/10.1007/s11042-022-13671-9 (2023).
29. Merzban, M. H. & Elbayoumi, M. Efficient solution of Otsu multilevel image thresholding: A comparative study. *Expert Syst. Appl.* **116**, 299–309. https://doi.org/10.1016/j.eswa.2018.09.008 (2019).
30. Kurban, R., Durmus, A. & Karakose, E. A comparison of novel metaheuristic algorithms on color aerial image multilevel thresholding. *Eng. Appl. Artif. Intell.* **105**, 104410. https://doi.org/10.1016/j.engappai.2021.104410 (2021).
31. He, L. & Huang, S. An efficient krill herd algorithm for color image multilevel thresholding segmentation problem. *Appl. Soft Comput.* **89**, 106063. https://doi.org/10.1016/j.asoc.2020.106063 (2020).
32. Martin, D., Fowlkes, C., Tal, D. & Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: *Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, 416–423. (IEEE, Vancouver, BC, Canada, 2001). https://doi.org/10.1109/ICCV.2001.937655
33. Sara, U., Akter, M. & Uddin, M. S. Image quality assessment through FSIM, SSIM, MSE and PSNR-a comparative study. *J. Comput. Commun.* **7**, 8–18. https://doi.org/10.4236/jcc.2019.73002 (2019).
34. Akay, B. & Karaboga, D. Solving integer programming problems by using artificial bee colony algorithm. In: *AI*IA 2009: Emergent Perspectives in Artificial Intelligence. Lect. Notes Comput. Sci.,* Vol. 5883 (eds Serra, R., Cucchiara, R.) 355–364 (Springer, Berlin, Heidelberg, 2009)
35. Derrac, J., García, S., Molina, D. & Herrera, F. 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 (2011).
36. Corominas, A. On deciding when to stop metaheuristics: Properties, rules and termination conditions. *Oper. Res. Perspect.* **10**, 100283. https://doi.org/10.1016/j.orp.2023.100283 (2023).
37. Essaid, M., Idoumghar, L., Lepagnot, J. & Brévilliers, M. Gpu parallelization strategies for metaheuristics: A survey. *Int. J. Parallel Emerg. Distrib. Syst.* **34**, 497–522. https://doi.org/10.1080/17445760.2018.1428969 (2018).
38. Salinas-Gutiérrez, R. & Zavala, A. E. M. An explicit exploration strategy for evolutionary algorithms. *Appl. Soft Comput.* **140**, 110230. https://doi.org/10.1016/j.asoc.2023.110230 (2023).
39. Zamani, H., Nadimi-Shahraki, M. H. & Gandomi, A. H. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.* **392**, 114616. https://doi.org/10.1016/j.cma.2022.114616 (2022).

Author contributions

I.B.: conceptualization, methodology, formal analysis, investigation, visualization, and writing—original draft. J.I.: methodology, formal analysis, validation, writing—review and editing, and supervision. All authors read and approved the final manuscript.

Funding

No funding was received for conducting this study.

Declarations

Competing interests

The authors declare that they have no conflict of interest.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable as the work is carried out on a publicly available dataset.

Additional information

Correspondence and requests for materials should be addressed to I.B.