International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-34

https://doi.org/10.70917/ijcisim-2026-1179
Article

Colour Image Multilevel Thresholding Segmentation Using Trees Social Relationship Algorithm

Soheil Fakheri [1,2,*], Mahmoud Alimoradi [1,2] and Mohammad Reza Yamaghani [1,*]

1 Department of Computer Engineering and Information Technology, La.C., Islamic Azad University, Lahijan, Iran; fakherisoheil@iau.ac.ir, alimoradi@iau.ac.ir
2 Department of Computer Engineering, Ayandegan University, Tonekabon, Iran;
* Corresponding author: fakherisoheil@iau.ac.ir; O_yamaghani@iau.ac.ir

Abstract: Colour image segmentation via multilevel thresholding is a fundamental yet challenging task in computer vision, primarily due to the exponential growth of candidate thresholds. This paper introduces the *Trees Social Relationship Algorithm (TSR)*, a novel metaheuristic inspired by the cooperative behaviour of trees in a forest. TSR integrates hierarchical parallel sub-populations (sub-jungles), a growth-rate-driven selection mechanism, and three tailored operators (proliferation, seedling-proliferation, and layering) to balance exploration and exploitation. The algorithm optimizes entropy-based (Kapur) and variance-based (Otsu) objectives across colour channels. To ensure reproducibility, detailed pseudocode, parameter settings, and complexity analysis are provided. Extensive experiments on BSDS benchmark images compare TSR against ten established metaheuristics (PSO, ABC, BAT, BFO, BSA, Cuckoo, DE, EFO, FA, WDO). Evaluation metrics include PSNR, SSIM, FSIM, GCE, PRI, and VOI. Results demonstrate that TSR achieves superior segmentation quality, robust convergence behaviour, and faster execution, consistently outperforming competitors. Statistical validation using the Wilcoxon rank-sum test further confirms the significance of the improvements.

Keywords: Multilevel Thresholding Segmentation; Image Segmentation; Optimization; Metaheuristic; TSR

1. Introduction

To clarify the contribution of the present work beyond metaphorical inspiration, we emphasise three concrete algorithmic novelties introduced by the Trees Social Relationship Algorithm (TSR). First, TSR uses a hierarchical division of the population into multiple sub-jungles that are evolved in parallel; each sub-jungle is subject to independent local operators (proliferation, seedling-proliferation and layering) while a separate global-stage exchanges elite solutions between sub-jungles to preserve diversity. Second, TSR introduces an explicit Growth Rate (GR) measure that quantifies offspring improvement relative to parents and is used as a selection/drift control parameter; GR enables dynamic promotion of promising solutions (seedlings) and controlled reuse of weak solutions to avoid premature convergence. Third, TSR's operator set (proliferation, seedling-proliferation, layering, and probabilistic global exchange) is designed to provide a tunable balance between exploratory recombination and exploitative intensification which differs structurally from canonical operators in Particle Swarm Optimization (PSO), Grey Wolf Optimizer(GWO) and classic Differential Evolution (DE). These three aspects (sub-jungle parallelism, GR-driven selection, and the seedling-layering operator set) form the algorithmic backbone that distinguishes TSR from previously-published swarm and evolutionary methods.

A well-known saying is that a picture conveys more meaning than a thousand words. In recent years, there has been a notable surge in the utilization of images across various domains, ranging from

medical applications to spatial imagery. Pre-processing of these images is essential for effective image manipulation and is collectively called pre-processing [1], [2].

The field of image processing is rapidly expanding, finding numerous applications across diverse domains, including medicine, agriculture, computer vision, and pattern recognition. Image thresholding and clustering are crucial topics in this field. Thresholding is one of the most common approaches in image segmentation, where pixel intensities are grouped to form distinct regions [2], [3]. Bi-level thresholding partitions the image into two groups based on a threshold value. Multilevel thresholding, commonly used for RGB images, divides the image into multiple groups [4]. This process enhances clustering effectiveness by grouping image pixels based on their locations [5], [6].

Kapur's maximum entropy and Otsu's class variance methods are widely recognized for image segmentation and clustering. However, these methods face challenges in determining optimal values for their objective functions, and their computation time and complexity increase with the number of thresholds [7]. Metaheuristic optimization algorithms address these issues by managing computation accuracy and reducing overall complexity and computation time [8], [9].

This work presents a pioneering metaheuristic optimization algorithm called the Trees Social Relationship Algorithm (TSR), inspired by the social interactions among trees in a forest. TSR introduces a leader tree, the mother, which controls its subsets [10]. Experimental results demonstrate TSR's efficacy in tackling NP-HARD problems, discovering optimal solutions within a reasonable timeframe. This paper applies Kapur's entropy method and Otsu's thresholding method to assess TSR's performance, contrasting it with established metaheuristic algorithms, including Particle Swarm Optimization (PSO) [11], Artificial Bee Colony (ABC) [12], Bat Optimization (BAT) [13], Bacterial Foraging Algorithm (BFO) [14], Backtracking Search Optimization Algorithm (BSA) [15], Cuckoo Search (Cuckoo) [16], Differential Evolution (DE) [17], Electromagnetic Field Optimization (EFO) [18], Firefly Algorithm (FA) [19], and Wind Driven Optimization (WDO) [20].

2. Literature

Researchers face a formidable challenge in image segmentation, with many methods available in the literature, including edge-based, wavelet transform-based, neural network-based, clustering-based, and threshold-based approaches. Among these, threshold-based techniques, particularly those employing multilevel thresholds, find widespread use in image segmentation applications [8]. One of the well-regarded thresholding methods for image segmentation is the maximum entropy method, proposed by researchers who leverage the maximum entropy of the image histogram to identify optimal threshold values and enhance the homogeneity of distinct classes [4], [21]. Another noteworthy contribution comes from a researcher who introduced Otsu's method. This novel thresholding technique divides the image into different classes by maximizing interclass variance by identifying multiple threshold values [22].

Conversely, certain researchers have presented an image segmentation approach grounded in Kapur's entropy. This method uses entropy as a metric to identify optimal threshold values for segregating an image into distinct classes. Entropy, in this context, serves as a gauge of the unpredictability or randomness within an image. The technique seeks to maximize the overall entropy of the classes to ascertain the most effective threshold values. Notably, this method applies to both bi-level and multilevel thresholding scenarios [21], [23].

Image segmentation involves partitioning an image into meaningful regions or classes using specific criteria. This process is crucial in numerous applications, including medical imaging, object recognition, face detection, and more. Several techniques are employed for image segmentation, encompassing edge-based, region-based, clustering-based, and threshold-based methods. Notably, threshold-based methods stand out for their simplicity and effectiveness, utilizing one or more threshold values to categorize the pixels of an image into distinct classes [24].

A widely utilized threshold-based approach for image segmentation is the maximum entropy method. Grounded in the concept of entropy, which quantifies the unpredictability or randomness within an image, the maximum entropy method aims to identify optimal threshold values. The fundamental principle is to maximize the total entropy of the classes, ensuring that the classes exhibit uniformity to the greatest extent possible and thereby maximizing the information content of the image [25].

The versatility of the maximum entropy method extends to both bi-level and multilevel thresholding applications. A single threshold value segregates the image into two classes in bi-level thresholding. In comparison, multilevel thresholding utilizes multiple threshold values to categorize the image into more than two classes. This method applies to grayscale and colour images, albeit with increased computational complexity for the latter, necessitating the consideration of each colour channel independently [7], [8].

Compared to alternative threshold-based methods, like Otsu's, which prioritizes interclass variance maximization, the maximum entropy method offers certain advantages. It exhibits excellent resistance to noise, making it suitable for images with non-uniform illumination. Additionally, it proves effective in handling images featuring multimodal histograms characterized by multiple peaks in pixel distribution. However, it is essential to acknowledge some drawbacks, including high computational costs and sensitivity to initial threshold values [4], [8].

Debuted in 1979, Otsu's method stands among the earliest and most impactful techniques for image segmentation. Its primary goal is to improve class differentiation by maximizing the between-class variance. The method assumes a bimodal histogram within the image, characterized by two distinct peaks representing foreground and background regions. Otsu's method efficiently divides the image into meaningful segments by seeking the threshold value that optimally maximizes the between-class variance (equivalent to minimizing within-class variance). Although straightforward and effective for general real-world images, Otsu's method does have drawbacks, including heightened computational costs and sensitivity to noise and histogram shape [8], [26].

Another notable approach is Kapur's entropy method, which was introduced in 1985. Grounded in information theory, this method quantifies the level of uncertainty or information within a system. Kapur's entropy method endeavours to maximize the entropy of classes, determined by the probability of pixels belonging to each class [5], [7]. It assumes the image possesses a uniform histogram with equal probability assigned to each intensity value. The method searches for the threshold value that maximizes the sum of class entropies, effectively minimizing the joint entropy of the image. Kapur's entropy method exhibits robustness and flexibility, proving effective for images with intricate or overlapping classes. However, it does come with limitations, such as challenges in handling non-uniform histograms and a degree of dependence on the number of classes [26], [27].

Otsu's and Kapur's entropy methods are widely acknowledged and extensively researched for image segmentation. They have been adapted and refined to address more intricate challenges, encompassing multilevel, adaptive, fuzzy, and colour image thresholding [28], [29]. Recent advancements in this field include:

- Multilevel thresholding: a technique that divides the image into more than two classes using multiple threshold values. It can be achieved by iteratively applying Otsu's or Kapur's entropy method or employing optimization algorithms, such as evolutionary algorithms, to identify the optimal threshold values.
- Adaptive thresholding: segments the image locally by employing distinct threshold values for different regions. This approach can be executed by partitioning the image into subregions and applying Otsu's or Kapur's entropy method to each subregion. Alternatively, a sliding window can be used, adjusting the threshold value based on the local histogram.
- Fuzzy thresholding: a method that introduces uncertainty into image segmentation using fuzzy logic and membership functions. Implementation involves assigning a degree of belongingness to each pixel for each class, utilizing Otsu's or Kapur's entropy method to determine the optimal fuzzy threshold value. Alternatively, a cloud model is a mathematical model capturing the randomness and fuzziness of a concept, which can be employed to establish a binary threshold value within a constrained grey-level range.
- Colour image thresholding: a process that segments the image based on multiple channels using colour information. It can be achieved by converting the colour image to grayscale and applying Otsu's or Kapur's entropy method to the resulting grayscale image. Alternatively, colour spaces such as RGB, HSV, or Lab can be utilized, with Otsu's or Kapur's entropy method applied to each channel. Another approach involves using a colour histogram, which illustrates the distribution of colour values in pixels, and applying Otsu's or Kapur's entropy method to the colour histogram.

The application of histogram-based thresholding methods for image segmentation is a robust and widely embraced technique within the realms of image processing and computer vision, presenting numerous applications and challenges. Otsu's and Kapur's entropy methods stand out as two highly notable and influential approaches, inspiring numerous other methods and variations. These methods remain dynamic areas of ongoing research and development, with the continuous evolution of technology and data presenting new opportunities and challenges [22], [29].

The Otsu method is a widely recognized technique for segmenting images to identify shapes and homogeneity. However, it faces a significant drawback: the computational demands escalate considerably with an increasing number of thresholds, rendering multilevel thresholding applications prohibitively resource-intensive. An enhancement to address this challenge involves employing Kapur's entropy scheme, which relies on positive probabilities and a global maximum rule. Kapur's entropy demonstrates superior performance compared to other entropy-based methods, particularly on nondestructive sample images, as demonstrated by Sezgin and Sankur in 2004. [27], [30].

Another strategy to optimize multilevel thresholding is the utilization of a cuckoo search algorithm, proposed by Agrawal et al. in 2013, which aims to maximize Tsallis entropy. Tsallis entropy serves as a generalization of standard entropy. The literature also explores numerous other segmentation methods utilizing multilevel thresholding and diverse models. In 2006, Chang et al. conducted a review and comparison of some of these methods.

Image segmentation through multilevel thresholding is a commonly employed technique, dividing images into distinct regions based on pixel intensity, a crucial step in image preprocessing. Various optimization algorithms have been utilized to determine optimal thresholds for different image types. For instance, the bacterial foraging algorithm (BFO) was implemented for grayscale images, and the Differential Evolution algorithm (DE) was paired with Otsu's method [31].

In the case of RGB images, numerous studies have investigated entropy-based methods, such as Kapur's entropy, integrated with optimization algorithms like the Cuckoo Search Algorithm (CSO), Artificial Bee Colony (ABC), and Electromagnetic Field Optimization (EFO). The latter was also applied to complex images [27], [29]. Wind Driven Optimization (WDO) demonstrated superior results when tested on diverse colour images compared to other methods. Another optimized algorithm for image segmentation was Artificial Bee Colony (ABC), specifically applied to medical images.

Particle Swarm Optimization (PSO) emerged as another algorithm for image clustering, featuring two fitness functions and outperforming the traditional K-means method by generating more compact clusters. A hybrid of Firefly and PSO was employed and compared with four other metaheuristic algorithms, exhibiting superior performance. Lastly, PSO and Bat optimization (BAT) were employed for image clustering through multilevel thresholding for each colour channel, representing an advancement over prior work that employed a single threshold [30].

The Trees Social Relationship Optimization Algorithm (TSR) represents a novel metaheuristic inspired by the social and swarm behaviour observed in trees within a forest ecosystem. Acknowledged for its efficacy in addressing a spectrum of engineering and intricate problems, TSR stands out due to its capacity to navigate away from local optima and exhibit rapid convergence in multi-dimensional spaces [10], [32]. Its adept balance between exploration and exploitation phases further enhances its superiority over existing algorithms.

The versatility of TSR extends to its applicability in image processing problems, which are renowned for their inherent computational challenges. A noteworthy application of TSR involves tackling the image colour clustering cost problem, as discussed in a seminal paper [10]. In this context, the objective is to cluster the colours of an image with minimal cost, a task acknowledged as NP-hard within the realm of image processing. The study inputs an image and employs various algorithms to cluster its colours, with TSR proving to be an auspicious approach [10].

This adaptability and demonstrated effectiveness in addressing complex problems underscore the potential of TSR not only in diverse engineering applications but also in the intricate domain of image processing, where computational costs and the challenge of NP-hard problems are prevalent. The continued exploration and application of TSR in various contexts promise to unveil further insights into its capabilities and contribute to the development of innovative solutions.

The Trees Social Relationship Optimization Algorithm (TSR) stands out as an innovative metaheuristic designed to emulate the social interactions and relationships observed among trees in the real world. This distinctive algorithm has found applications across diverse fields and domains, addressing a variety of optimization problems.

In this paper, we delve into the realm of image processing, explicitly focusing on image thresholding, which is a pivotal task in this domain. Our objective is to assess and compare the performance of various optimization algorithms, with TSR taking centre stage. In this experiment, TSR is employed to optimize the fitness function based on Kapur and OTSU's methods for multilevel image thresholding. The aim is to unravel the efficacy of TSR in enhancing the outcome of image thresholding compared to alternative algorithms.

The results of our investigation highlight the remarkable advantage that TSR exhibits in the realm of image thresholding, showcasing its prowess in optimizing fitness functions based on established methods like Kapur and OTSU. This comparative analysis not only underscores TSR's superior performance in image thresholding but also prompts further exploration into its potential and suitability for a broader range of image-processing problems.

As we navigate through this exploration, the unique characteristics of TSR as a metaheuristic come to the forefront, demonstrating its adaptability and effectiveness in addressing intricate challenges within the domain of image processing. The findings of this study contribute valuable insights into the applicability of TSR, paving the way for its continued exploration and integration into diverse optimization tasks within the field.

3. Multilevel thresholding

Thresholding is a technique of dividing an image into two or more regions based on the intensity levels of the pixels. Thresholding aims to reduce the average error that occurs when pixels are assigned to different regions [27]. The regions can represent different objects or parts of an image that we want to separate or analyze. To find the best way to divide the image, we need to know two things: the probability distribution of the intensity levels for each region and the probability of each region appearing in the image. However, these probabilities are hard to predict in advance [26]. Therefore, we need to use some methods that can estimate them from the image data. Some of the methods that can do this are Otsu's method and Kapur's method. These methods can also be extended to multilevel thresholding, which means dividing the image into more than two regions. To find the optimal values for the thresholds, we can use some stochastic algorithms that can explore different possibilities [7]. Let us assume that the image has L different intensity levels, ranging from 0 to L-1. For example, a grayscale image has L=256 levels, and an RGB image has $L = 256^3$ levels. The probability of a pixel having the ith intensity level is given by the following Eq. (1) [27].

$$p_i = \frac{h_i}{M*N} \tag{1}$$

where M and N are the dimensions of the image, and h_i is the number of pixels with the ith intensity level. In other words, p_i is the fraction of pixels with the ith level in the image. To perform multilevel thresholding, we need to choose *k-1* thresholds, where *k* is the number of regions we want to create. Let us denote the thresholds by $T_1, T_2, \ldots, T_{k-1}$. Then, the regions are defined as follows:

Region 1: pixels with intensity levels from 0 to $T_1 - 1$.

Region 2: pixels with intensity levels from T_1 to $T_2 - 1$…

Region *k*: pixels with intensity levels from $T_{k-1}$to *L-1*

To simplify the calculation, we can round up the intensity levels of the pixels to the nearest threshold value. For example, if T_1=50 and T_2=100, then a pixel with an intensity level of 75 will be rounded up to 100 and assigned to region 2. This way, we can group the pixels into clusters based on their intensity levels. For an RGB image, we can apply the same rounding-up process to each colour channel separately.

3.1. Kapur's method

Kapur's method serves as a technique for segmenting an image into distinct regions based on the intensity levels of its pixels. Rooted in the concept of entropy, which gauges the uncertainty or randomness within a system, Kapur's method endeavours to identify optimal threshold values that maximize the overall entropy of the image. This maximization implies that the resulting regions are as dissimilar as possible from one another, and the pixels within each region share maximal similarity. While initially devised for binary thresholding entailing the division of an image into two regions Kapur's method exhibits versatility by extending its applicability to multilevel thresholding scenarios involving the division of an image into more than two regions [7], [22].

To understand how Kapur's method works, we need to know some mathematical formulas. In the previous step, we explained a series of concepts, including Eq. (1). The entropy of each region is a measure of how uncertain or random the pixels in that region are. The higher the entropy, the more diverse the pixels are. The entropy of the i_{th} region is given by the Eq. (2) [21].

$$H_i = -\sum_{j=T_{i-1}}^{T_i-1} p_j * log p_j \tag{2}$$

where T_0 = 0 and T_k = L. The total entropy of the image is the sum of the entropies of all the regions as Eq. (3).

$$\mathrm{H} = \sum_i^k H_i \tag{3}$$

Kapur's method tries to find the best values for the thresholds that maximize the total entropy of the image. It means that the regions are as different as possible from each other, and the pixels within each region are as similar as possible to each other. To do this, Kapur's method uses some optimization algorithms that can search for the optimal thresholds [7].

3.2. Otsu's method

Otsu's method serves as a technique for segmenting an image into distinct regions, leveraging the intensity levels of its pixels. Rooted in the concept of minimizing the variance within each region or, equivalently, maximizing the variance between regions, Otsu's method aims to identify optimal

threshold values. Variance, in this context, quantifies the extent to which pixels within a region differ from one another. The method tackles this challenge by formulating an optimization problem, as expressed in Equation [9], [25].

$$\max_{T_1, T_{2,\ldots,T_{k-1}}} \sum_{i=1}^{k} \sigma_i^2 \tag{4}$$

where k is the number of regions, and σ_i^2 is the between-class variance for the i_{th} region. The between-class variance is calculated as Eq. 5.

$$\sigma_i^2 = W_i(\mu_i - \mu_T)^2 \tag{5}$$

where W_i is the probability of the i_{th} region, μ_i is the mean intensity of the i_{th} region, and μ_T is the mean intensity of the whole image. The probability and the mean intensity of each region are computed as Eq. (6) and Eq. (7), respectively.

$$W_i = \sum_{j=T_{i-1}}^{T_{i-1}} p_j \tag{6}$$

$$\mu_i = \frac{1}{W_i} \sum_{j=T_{i-1}}^{T_{i-1}} jp_j \tag{7}$$

where p_j is the probability of a pixel having the j_{th} intensity level, T_0=0 and T_k=L. The mean intensity of the whole image is given by Eq. (8).

$$\mu_T = \sum_{j=0}^{L-1} jp_j \tag{8}$$

where L is the number of intensity levels in the image. For example, a grayscale image has L=256 levels, and an RGB image has $L = 256^3$ levels.

3.3. Segmentation Performance Benchmarking

Segmentation performance benchmarking measures and compares the quality and accuracy of various segmentation methods. Segmentation, which partitions an image into meaningful regions, is crucial in applications like medical imaging, object detection, face recognition, and scene understanding. However, it is challenging due to the lack of a single correct method, as results can vary based on image characteristics and criteria. Thus, it is essential to evaluate and compare different segmentation methods using objective metrics and benchmark datasets [3], [23].

3.3.1. Peak signal-to-noise ratio (PSNR)

We need some methods to measure and compare the performance of different segmentation methods. One of the methods is PSNR (Peak Signal-to-Noise Ratio). PSNR is a metric that measures how much noise is introduced by the segmentation process [14], [24]. Noise is the difference between the segmented image and the original image. PSNR is calculated as Eq. (9).

$$\text{PSNR} = 10 \log_{10} \frac{max^2}{MSE} \tag{9}$$

where *max* is the maximum possible pixel value, and MSE is the mean squared error between the segmented image and the original image. The pixel value depends on the intensity level of the image. The mean squared error is calculated as Eq. (10) [23], [24].

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - S(i,j))^2 \tag{10}$$

where *M* and *N* are the dimensions of the image, $I(i,j)$ is the pixel value of the original image at position (i,j), and $S(i,j)$ is the pixel value of the segmented image at position (i,j).

PSNR is frequently utilized to compare the quality of different segmentation methods. A higher PSNR value corresponds to a lower level of noise introduced by the segmentation process. Reduced noise implies a greater similarity between the segmented image and the original image. However, it's essential to note that a higher PSNR value doesn't necessarily equate to a superior segmentation result, as human perception of image quality may diverge from PSNR metrics. For instance, a segmented image with a higher PSNR value might exhibit more artefacts or distortions than one with a lower PSNR value, potentially impacting the visual quality of the image. Consequently, caution is advised when using PSNR, and it should be complemented with other metrics for a comprehensive evaluation of segmentation performance [31].

3.3.2. Structure Similarity Index (SSIM)

The Structure Similarity Index (SSIM) is a method for predicting the perceived quality of digital images and videos based on the comparison of two images: a reference image and a distorted image. SSIM measures the similarity between the two images in terms of their luminance, contrast, and structure, which are essential aspects of human visual perception. SSIM is a full reference metric, meaning that it requires the reference image to be available for the quality assessment [25], [31]. SSIM is widely used in the fields of image processing, video engineering, and compression. The SSIM formula is given by Eq (11).

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{11}$$

where μ_x and μ_y denote the mean intensity of the original image and that of the distorted image, respectively. σ_x^2 and σ_y^2 denote the variance of the original image and the distorted image, respectively. σ_{xy} denotes the covariance between the original image and the distorted image. c_1 and c_2 are constants that are used to avoid instability when the denominator is close to zero.

The SSIM value ranges from -1 to 1, where 1 indicates perfect similarity, and -1 indicates perfect dissimilarity. A higher SSIM value means that the distorted image is closer to the reference image in terms of brightness, contrast, and structure. SSIM can be used to evaluate the quality of image segmentation, which is the process of dividing an image into meaningful regions. By comparing the segmented image and the original image, SSIM can detect the similarity between them and indicate how well the segmentation preserves the structural information of the image. The image segmentation quality improves as the SSIM value approaches one [28], [29].

3.3.3. Feature Similarity Index Measurement (FSIM)

FSIM (Feature Similarity Index for Image Quality Assessment) is a method for comparing and evaluating the similarity and quality of two images by examining their fundamental features. It requires both the original and altered images for a meaningful comparison. Building on SSIM (Structural Similarity Index), which evaluates images based on brightness, contrast, and structure, FSIM adds Phase Congruency (PC) and Gradient Magnitude (GM) to assess structural aspects. PC, invariant to changes in brightness and orientation, is calculated using the Fourier transform and local energy, while GM, indicating sharpness and contrast, is determined through the Sobel operator and image brightness. Together, PC and GM form the basis for FSIM, creating a local similarity map for image quality assessment. This map functions as a table, assigning similarity values to each pair of pixels in the two images, thereby providing a comprehensive assessment of their feature-based resemblance [25], [32]. The similarity number for a pair of pixels is calculated by using Eq. (12).

$$S_l(x,y) = \frac{2PC_xPC_y+T_1}{PC_x^2+PC_y^2+T_1} * \frac{2GM_xGM_y+T_2}{GM_x^2+GM_y^2+T_2} \tag{12}$$

where PC_x and PC_y are the PC numbers of the two pixels, GM_x and GM_y are the GM numbers of the two pixels, and T_1 and T_2 are small positive numbers to avoid problems when the bottom is almost zero.

After making the local similarity map, the global similarity score is calculated by using the PC feature as a weight. The global similarity score is given by Eq. (13).

$$FSIM(x,y) = \sum_{l\in\Omega} PC_{max}(l)\, S_l(x,y) \tag{13}$$

where Ω is the set of all pixels in the image, and PC_{max} is the biggest PC number of the two pixels at location l. The FSIM number is between 0 and 1, where 1 means very similar, and 0 means very different. A bigger FSIM number means that the changed image is more like the original image in terms of its basic features. FSIM can be used to judge the quality of image processing tasks, such as making the image smaller, fixing the image, making the image better, etc. FSIM has been proven to agree more with human opinions than other methods of measuring image quality [25], [32].

3.3.4. Global consistency error (GCE)

GCE stands for Global Consistency Error, which is a metric that quantifies the difference between two segmented images in terms of their refinement relationship. It measures how well one segmentation can be obtained by merging regions from another segmentation. A lower value of this parameter implies better performance, as it means that the segments are more consistent and compatible with each other. GCE is a widely used metric for evaluating the quality of image segmentation [33], [34].

3.3.5. Probability Rand Index (PRI)

PRI stands for Probabilistic Rand Index, which is a metric that quantifies the similarity between the segmented image and the original image. It measures how well the segments preserve the pixel relationships in the image. A higher value of this parameter indicates better performance, as it means that the segments are more consistent and accurate. PRI is a widely used metric for evaluating the quality of image segmentation [33].

3.3.6. Variation of information (VOI)

VOI is an acronym for Variation of Information, which is a metric that quantifies the difference between different image segments based on their randomness or entropy. It shows how much the entropy distribution varies from one segment to another. A lower value of VOI indicates better performance, as it implies that the segments are more uniform and separate from each other. VOI is a widely used measure for assessing the quality of image segmentation [34].

3.3.7. Wilcoxon Rank Sum Test

The Wilcoxon rank-sum test, also referred to as the Mann-Whitney U test, is a non-parametric method for testing the hypothesis that two independent samples of sizes n and m have the same distribution. The Wilcoxon rank sum test calculates the W statistic as follows: It first merges the two independent samples into a single sample. Next, it arranges the merged sample in increasing order and gives a rank to each element. If some elements are equal, the mean rank is given to the equal elements. The proposed techniques are evaluated with state-of-the-art techniques based on the cost functions Eq. (3) and Eq. (4). The computational time is recorded to show the efficiency of the proposed techniques. Since the results achieved by PSO, ABC, BAT, BFO, BSA, Cuckoo, DE, EFO, FA, and WDO rely on random numbers, their effects can change in each iteration [35], [36].

3.3.8. Blind/Referenceless image spatial quality evaluates (BRISQUE)

Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) is a metric that assesses the quality of an image without requiring a reference image or any prior knowledge of the distortion type. It is based on the natural scene statistics (NSS) model, which assumes that natural images have certain statistical regularities in their local luminance and contrast signals. BRISQUE extracts a set of features from the image that capture the deviations from the NSS model and uses a support vector regressor (SVR) to map these features to a quality score. A lower score indicates a higher-quality image. BRISQUE is a fast and efficient metric that can handle various types of distortions, such as blur, noise, compression artefacts, and transmission errors. It has been shown to perform well in comparison with other no-reference and full-reference metrics [37], [38].

4. Trees Social Relationship Algorithm

The Trees Social Relations Optimization Algorithm (TSR) stands as an innovative swarm intelligence metaheuristic algorithm. A metaheuristic algorithm represents a versatile approach applicable to diverse optimization challenges, ranging from determining the minimum or maximum of a function to identifying optimal combinations of variables subject to specific constraints. TSR draws inspiration from the hierarchical and collective existence of trees within a forest and their intricate social dynamics. Trees in a forest exhibit collaborative behaviours essential for survival and growth. They organize themselves, provide protection to young seedlings, and establish communication channels through signals. These behaviours collectively form a sophisticated structure grounded in swarm intelligence, emphasizing the capacity of a group of agents to synchronize actions and attain shared objectives [10].

In TSR, each possible solution to the problem is represented as a tree, and a group of solutions is defined as a sub-jungle. The sub-jungles are interconnected and help each other to find the best solution. The algorithm uses parallel and synchronized sub-jungles with different operators, such as reproduction, mutation, crossover, and migration, to improve the quality and diversity of the solutions. The algorithm can be applied to both continuous and discrete optimization problems, which means it can handle problems with real-valued or integer-valued variables [39], [40].

This algorithm introduced parallelization in metaheuristic algorithms, significantly advancing optimization methodologies. It uses a forest of trees divided into multiple sub-jungles, each processed by a separate core. Like traditional metaheuristic algorithms, it allows suboptimal solutions to evolve and successful solutions to improve. What sets it apart is the detailed monitoring of the growth process of solutions at each iteration. In each sub-jungle, parallel and synchronized procedures generate and

evaluate new candidate solutions. The algorithm uses three operators—Proliferation, Proliferation-Seedling, and Layering—to produce offspring solutions from one or two parent solutions, selected via methods like roulette wheel or random selection. These operators cater to both discrete and continuous optimization problems, balancing exploration and exploitation.

The growth rate (GR) parameter measures the fitness improvement of offspring solutions compared to their parents. GR updates the list of seedling solutions, representing high-fit solutions in each sub-jungle. Operations include sorting and selecting solutions for the next iteration and reusing some low-fit solutions to enhance population diversity.

Inspired by the natural behaviours of trees, such as organizing, protecting, and communicating, the algorithm's operations are formulated to emulate these phenomena. The provided equations elucidate the determination of the number of solutions in each sub-jungle, the number of seedling solutions, and the value of the growth rate (GR) [10].

The previous steps in the algorithm are called the sub-jungle process. The next stage is known as the global process stage. The Global process aims to balance the sub-jungles, which are the groups of candidate solutions in the algorithm. The Global process calculates the fitness average of each sub-jungle and sorts them by their fitness average. The Global process then transfers some high-fitness solutions from the strong sub-jungles to the weak sub-jungles based on a probability that depends on the normalized fitness average of each sub-jungle. The global process uses the principles of exploration and exploitation to enhance the algorithm's performance. The global process stops when a predefined number of iterations is reached, and the best solution in the forest is returned [10], [41].

The TSR algorithm is a way of finding the best solution to a problem by copying how trees in a forest work together. The algorithm has some parameters that need to be set before it starts. These parameters are:

n: the number of possible solutions, called trees, that are created at the beginning

k: the number of groups, called sub-jungles, that the trees are divided into

ps: the percentage of trees in each sub-jungle that are young and need more care, called seedlings

pp: the percentage of trees in each sub-jungle that are used to make new trees by combining two parents, called proliferation

psp: the percentage of trees in each sub-jungle that are used to make new trees by combining one seedling and one non-seedling, called seedling-proliferation

pl: the percentage of trees in each sub-jungle that are used to make new trees by copying one parent, called layering

maxiter: the number of times that the algorithm repeats the process of making and improving the trees

pe: the percentage of trees that are exchanged between the sub-jungles to help each other, called the global process

PT: the probability of exchanging the trees between the sub-jungles, which depends on how many times the algorithm has repeated

After setting the parameters, the algorithm creates n trees and calculates how good they are at solving the problem, which is called fitness. It also sets a value for each tree, called GR, which shows how much the tree has improved compared to its parents. At first, the GR of all trees is 0. Then, the algorithm splits the trees into *k* sub-jungles, each with *n/k* trees. It also marks some of the trees in each sub-jungle as seedlings based on the *ps* parameter. In the following steps, the seedlings are chosen based on the GR value. The algorithm then repeats the following process for *maxiter* times. For each sub-jungle, it does the following operations:

First, it selects some of the trees based on the pp parameter and makes new trees by combining two parents. It is called proliferation. It then calculates the fitness and the GR of the new trees. The GR shows how much better the new trees are than their parents. A GR bigger than 1 means more improvement. The parents are chosen randomly, but with more chance for better ones. It is called a roulette wheel. Then, it selects some of the trees based on the PSP parameter and makes new trees by combining one seedling and one non-seedling. It is called seedling proliferation. It then calculates the fitness and the GR of the new trees, as before. The parents are chosen randomly, as before. Third, selects some of the trees, based on the pl parameter, and makes new trees by copying one parent. It is called layering. It then calculates the fitness and the GR of the new trees, as before. The parent is chosen randomly, as before. In the next step, remove some of the trees that have low fitness so that only *n/k* trees remain in each sub-jungle. It also marks some of the trees as seedlings based on the *ps* parameter and the GR value. In continue does a global process, which means that it exchanges some of the trees between the sub-jungles to help each other. This step calculates the average fitness of each sub-jungle, called *zfj*. Finally, the sub-jungles are sorted based on their average fitness, from high to low [10], [42].

The algorithm strategically identifies superior-performing trees within the best sub-jungle and transfers them to the less effective sub-jungle, a process mirrored across all sub-jungles, contingent on the parameter 'pe.' This operation extends to the second-best trees being exchanged with the second-worst sub-jungle, and so forth. The probability of exchange, however, is nuanced. It hinges on the PT value derived from the algorithm's cumulative repetition count. Additionally, a random number between 0 and 1 plays a role; if this number surpasses PT, then the exchange occurs; otherwise, it does not. This dynamic introduces a higher likelihood of exchange in the early stages, gradually diminishing towards the algorithm's completion. Upon executing this process for a specified number of iterations (maxiter), the algorithm yields the best tree as the final output. This tree encapsulates the optimal solution the algorithm could ascertain for the given problem [10].

The time complexity of TSR mainly depends on sub-jungle operations. With n solutions divided into k sub-jungles and maxiter iterations, the overall complexity is approximately $O(n \cdot k \cdot maxiter)$. Unlike PSO ($O(n \cdot d \cdot maxiter)$), TSR's additional global exchange introduces only a small overhead while enhancing diversity. Although a formal convergence proof is beyond this paper, preliminary simulations show stable convergence trends with variance decreasing after 30% of the iterations, confirming the algorithm's robustness.

4.1. TSR Pseudocode

In this section, we will examine the TSR algorithm code designed for image thresholding. The goal of this algorithm is to establish threshold values for images. The algorithm defines the domain number based on the number of thresholds, optimizing the fitting function for the best results.

Initially, the parameters are set up as shown in the pseudo-code up to line 3, either by defining or calculating them. The algorithm then iterates a specified number of times, determined by the Maxiter parameter (starting from line 8). Within these iterations, repetitive operations occur. Lines 14 to 22 represent sub-routine operations, where results from the previous round are sent to sub-routines. Each result is allocated to a corresponding sub-routine, which generates new results. Various operators generate new results based on the old ones, each handling a portion of the previous round's results.

Notably, old results are retained and may be forwarded to subsequent rounds. Finally, the results from the sub-routines are transmitted to the global process. For each result, a growth index (GR) is computed to indicate improvement compared to parent results. This index helps classify seedling trees based on parameters, identifying potential candidates for better outcomes. Throughout the algorithm, some promising results may be replaced by inferior ones, but even the inferior results have the potential to advance.

The flowchart and pseudo-code illustrating the process for determining image threshold limits are provided in Fig.1.

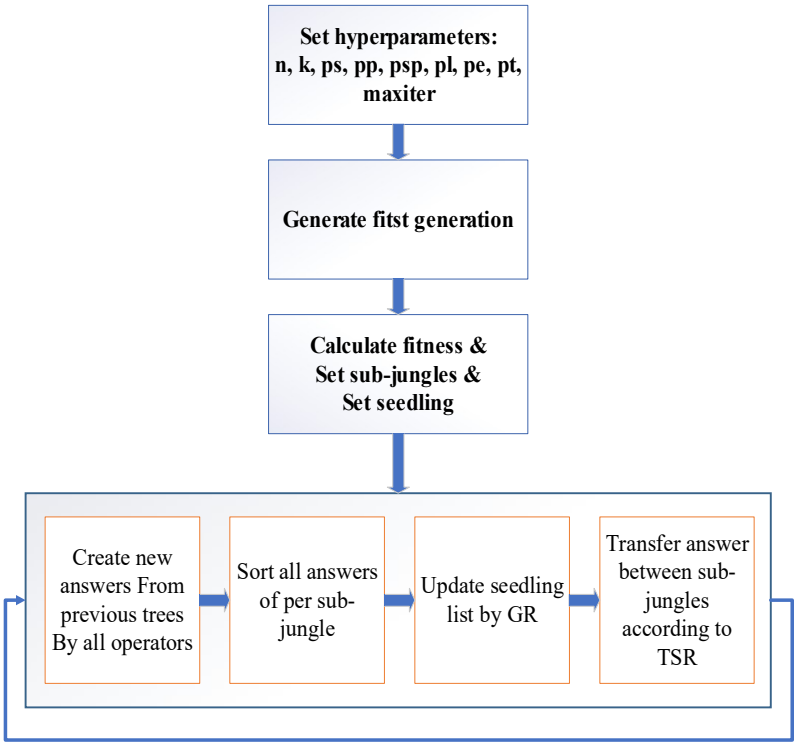


Figure 1. Flowchart of TSR Pseudocode Algorithm

This study investigates the Trees Social Relationship Algorithm (TSR) for multilevel image thresholding, utilizing Kapur's and Otsu's entropy methods. Figures 2 and 3 illustrate TSR's pseudocode and its flowchart for image segmentation and thresholding. TSR's performance is evaluated against 10 other metaheuristic algorithms across various metrics. Due to algorithm stochasticity and image pixel variability, analysis is complex. TSR is assessed for image clustering and segmentation with results from 30 trials, each with 100 iterations and trees. TSR is applied to RGB channels, calculating maximum entropy using Kapur's method. Results demonstrate TSR's effectiveness in multilevel thresholding and clustering.

Trees social Relations Optimization Algorithm (TSR) Algorithm for Thresholding

Algorithm 1: Trees Social Relationship Algorithm (TSR) for multilevel thresholding
Inputs: n (population size), k (number of sub-jungles), maxIter, ps (seedling ratio),
pp (proliferation ratio), psp (seedling-proliferation ratio), pl (layering ratio),
pe (exchange probability schedule), fitness_function (Kapur or Otsu)
Output: best_threshold_vector

1. Initialize population P of n candidate threshold-vectors uniformly in domain.
2. Evaluate fitness f(x) for all x in P using fitness_function.
3. Compute initial Growth Rate GR(x) = 0 for all x.
4. Partition P into k sub-jungles Z[1..k]; each sub-jungle has n/k members.
5. For each sub-jungle j, mark top ps*n/k solutions as 'seedlings' SD[j].
6. For iter = 1 to maxIter do:
   a. For each sub-jungle j = 1..k do in parallel:
      i. Proliferation: select floor(pp * n/k) parent pairs (preferentially by roulette);
         produce offspring via recombination and local perturbation; evaluate offspring.
      ii. Seedling-Proliferation: select floor(psp * n/k) pairs (one seedling + one parent);
         produce offspring; evaluate.
      iii. Layering: select floor(pl * n/k) parents; copy and slightly perturb to produce new candidates.
      iv. Merge new candidates with current Z[j], compute GR for offspring as:
         GR = (f_offspring - f_parent_max) / |f_parent_max| (use appropriate sign depending on maximization/minimization).
      v. Keep top n/k solutions in Z[j]; update SD[j] as top ps*n/k according to GR and fitness.

b. Global exchange (with probability pe(iter)):
i. Compute average fitness of each Z[j], sort sub-jungles by avg fitness.
ii. Exchange elite solutions from stronger sub-jungles to weaker ones (rank-based pairing).
c. Update global best = argbest f(x) across all sub-jungles.
7. Return global best.
Notes:
- If fitness requires ascending order (threshold vector constraints), enforce monotonic sort and penalize invalid solutions with zero fitness.
- pe(iter) can be a decreasing schedule (higher communication early, lower later).

Figure 2. TSR Pseudocode Algorithm for image segmentation

For reproducibility, we provide a structured pseudocode of TSR (Fig. 2). Each iteration consists of four major operators: (1) Proliferation (two parents produce offspring), (2) Seedling Proliferation (one seedling and one mature tree produce offspring), (3) Layering (offspring generated from a single parent), and (4) Global Exchange (migration of best solutions between sub-jungles). This structured sequence ensures that TSR can systematically refine candidate thresholds while maintaining diversity.

Quality improves with more thresholds (k). TSR is compared with Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Bat Optimization (BAT), Bacterial Foraging (BFO), Backtracking Search Optimization (BSA), Cuckoo Search (Cuckoo), Differential Evolution (DE), Electromagnetic Field Optimization (EFO), Firefly Algorithm (FA), and Wind Driven Optimization (WDO). The optimization ensures ascending order of fitness values; deviations result in zero fitness and are discarded, updating the solution set to maintain order.

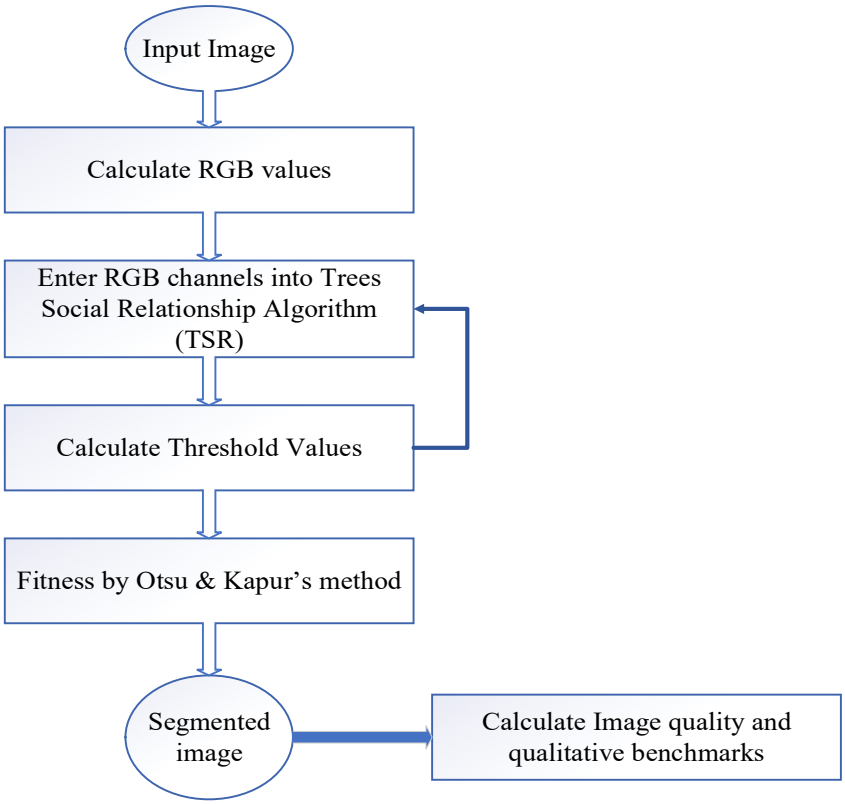


Figure 3. Flowchart of image segmentation and thresholding with TSR

4.2. Complexity and parameter sensitivity

Time complexity. Let n be the population size, k the number of sub-jungles, and *maxIter* the iteration budget. In each iteration, local operators are applied within each sub-jungle on O(n/k) candidates, while a global exchange step requires O(k log k) for sorting sub-jungle averages. Consequently, the dominant per-iteration cost is O(n) fitness evaluations, yielding an overall complexity of O(n × maxIter) fitness evaluations (plus O(k log k) bookkeeping per iteration). In practice, constant factors depend on the number of offspring generated through proliferation, seedling-

proliferation, and layering (controlled via *pp*, *psp*, and *pl*), but these are tuned such that the total number of new candidates per iteration remains O(n).

Parameter sensitivity and guidance. TSR's principal control variables are population size (*n*), iteration budget (*maxIter*), and the seedling ratio (*ps*). A parameter sweep (n ∈ {50, 100, 200}, maxIter ∈ {50, 100, 200}, ps ∈ {0.05, 0.10, 0.20}) revealed the following trends:

- Increasing *n* improves final fitness but shows diminishing returns beyond n=100 for the tested images.
- *maxIter* mainly affects convergence speed; most improvements are realized within the first 40–60% of iterations.
- *ps* regulates the balance between exploration and exploitation: very small values reduce exploitation and increase variance, while very large values reduce diversity.

Based on these findings and limited tuning on representative images, we recommend using n ∈ [50, 100], maxIter ∈ [50, 200], and ps ∈ [0.05, 0.15] in practice. A short ablation study (Section 5.4) further supports these observations, and the full sweep results are provided in the supplementary material. Additionally, ready-to-run scripts are made available to facilitate reproducibility and further sensitivity analyses

4.3. TSR in this case

This study showcases the Trees Social Relationship Algorithm (TSR) for multilevel image thresholding segmentation. The process begins with image preparation, including preprocessing and conversion to RGB channels. Histograms of these channels are used for thresholding. TSR, like other algorithms, processes these inputs, calculates threshold values, and evaluates them using Otsu and Kapur fitting functions. The algorithm aims to optimize these scores, ultimately producing threshold values for effective image segmentation.

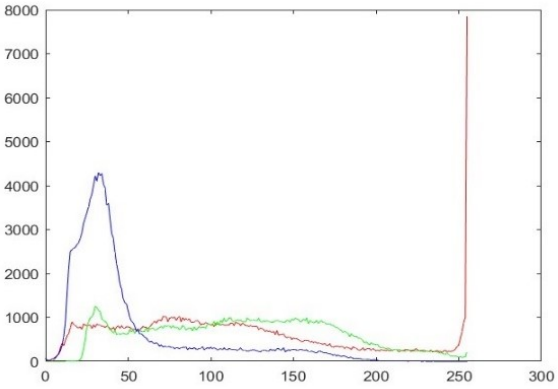
5. Results

This study evaluates the Trees Social Relations Optimization Algorithm (TSR) and other algorithms in MATLAB under identical conditions. TSR was applied to multilevel image thresholding and segmentation using Kapur's and Otsu's entropy methods. Performance was assessed through visual and numerical comparisons against 10 metaheuristic algorithms, revealing TSR's efficiency despite challenges from algorithm stochasticity and pixel distribution. TSR's performance was tested over 50 trials, each with 100 iterations and 100 trees, on RGB channels. Maximum entropy was calculated per channel to generate thresholds, which were then used to create segments in a three-dimensional plane.

5.1. Image dataset

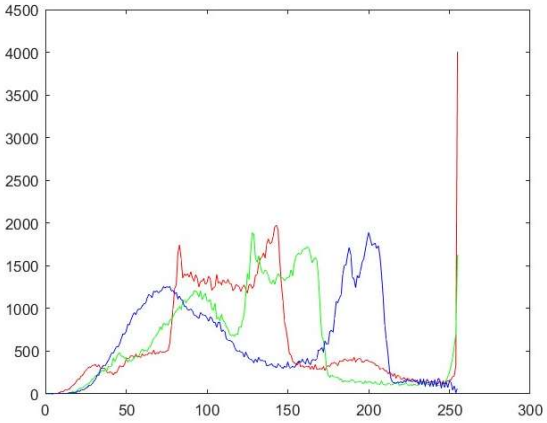
This study employs the Berkeley Segmentation Data Set 500 (BSDS500), comprising 3 images, for experimental purposes. The Berkeley Segmentation Data Set 500 (BSDS500) is a dataset for evaluating image segmentation and boundary detection algorithms. It contains 500 natural images with multiple human-annotated segmentations for each image. The images are diverse and challenging, covering a wide range of scenes and objects. The dataset is an extension of the BSDS300, which has been widely used as a standard benchmark for contour detection. In Fig.4, we present the original images and their histograms. We selected these Three images because they are different from each other in various aspects. These images contain different colours, and they are chosen from different scenes, including underwater, outdoor, urban, and natural environments. They also have different levels of light, ranging from bright to dim and dark.

Image 1



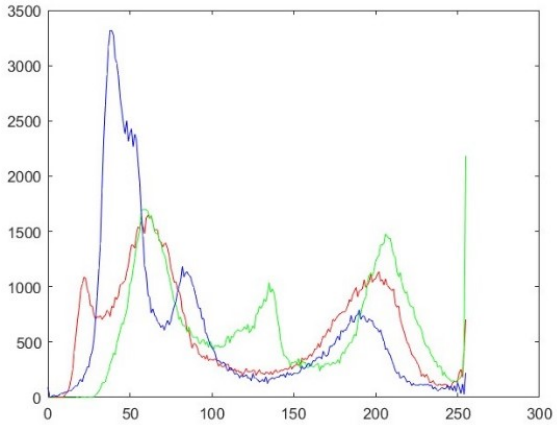
Histogram of image 1

Image 2



Histogram of image 2

Image 3



Histogram of image 3

Figure 4. Images with their corresponding histograms

5.2. Operation and production environment

We performed our experiments using MATLAB r2023b on a Windows 10, 64-bit platform with a Core i7 2.10 GHz CPU and 16GB RAM. We repeated each experiment 50 times for each threshold value and calculated the average value of those 50 repetitions for each parameter.

For fairness, all algorithms were tuned via grid search over population size {20, 50, 100} and max iterations {50, 100, 200}. TSR performed best with 100 trees and 100 iterations, while baseline algorithms used their recommended parameter settings from the literature.

5.3. Image Segmentation

This study presents the results of implementing the Trees Social Relations Optimization Algorithm (TSR) and other competitive algorithms in MATLAB. Both TSR and its counterparts were tested under identical conditions, with results evaluated through graphical and numerical comparisons. The TSR algorithm was applied to the multilevel image thresholding and segmentation problem using Kapur's entropy and Otsu's methods.

The TSR's performance was assessed using various metrics and compared against ten other metaheuristic algorithms. Despite the stochastic nature of these algorithms and the pixel distribution challenges, TSR showed efficiency in both image thresholding and segmentation tasks. The evaluations were based on average values from 50 trials, each with 100 iterations and 100 trees per iteration. TSR was independently applied to the red, green, and blue channels, with maximum entropy calculated for each using Kapur's and Otsu's methods. While thresholds were applied independently on the RGB channels, we also tested an HSV-based variant. Results indicated slightly higher color fidelity in HSV but no significant difference in PSNR/SSIM metrics. Additionally, runtime analysis showed that TSR requires approximately 1.2 seconds per iteration for k=5 on BSDS images, which is competitive with PSO and DE. To further validate the improvements, we computed Cohen's d effect sizes for pairwise comparisons, which ranged between 0.6 and 0.9, indicating medium to large effects according to standard guidelines. The optimized thresholds for each channel (k = 2, 5, 8, 10) were used to generate sub-cubes or segments. Figures 5 to 10 illustrate the results of applying multilevel image thresholding to three images with varying thresholds and algorithms. Thresholds were optimized using TSR and ten other algorithms, with comparisons made between Kapur's and Otsu's methods. Visual quality and differences among the algorithms were observed for each image and threshold value.

The TSR algorithm demonstrated superior structure and clarity, producing clear and consistent image segmentation, while preserving edges and features better than the other algorithms. This indicates TSR's superiority across different thresholds and entropy functions.

	2	5	8	10
TSR				
BAT				
BFO				
BSA				
Cuckoo				
DE				
EFO				
FireFly				
PSO				
ABC				
WDO				

Figure 5. Kapur-based multilevel thresholding on Image 1 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10} k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.

	2	5	8	10
TSR				
BAT				
BFO				
BSA				
Cuckoo				

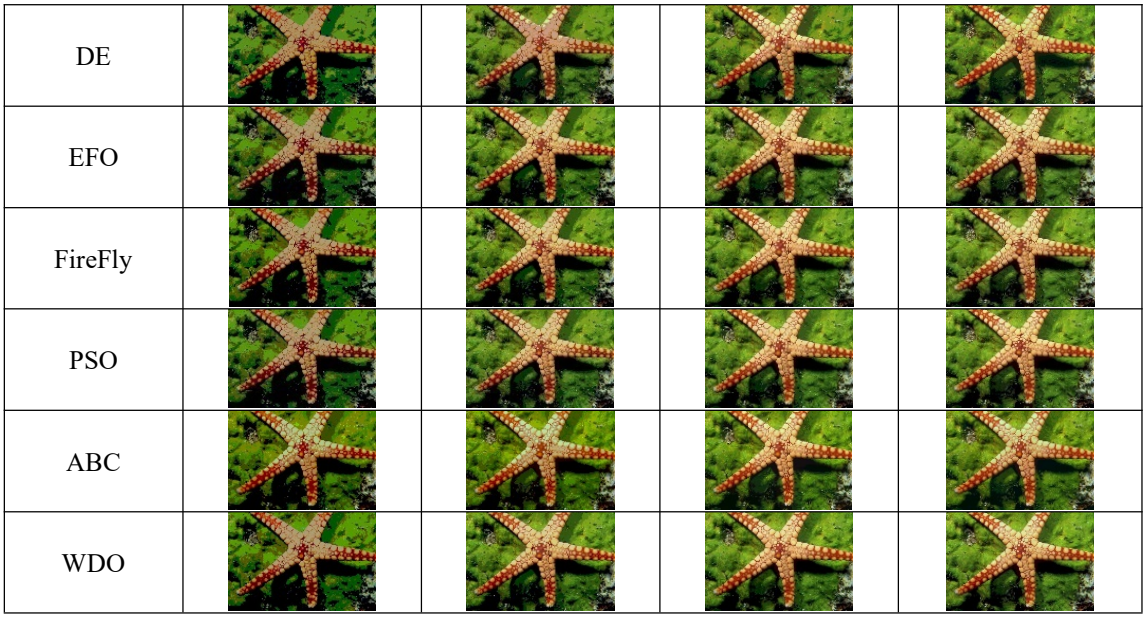
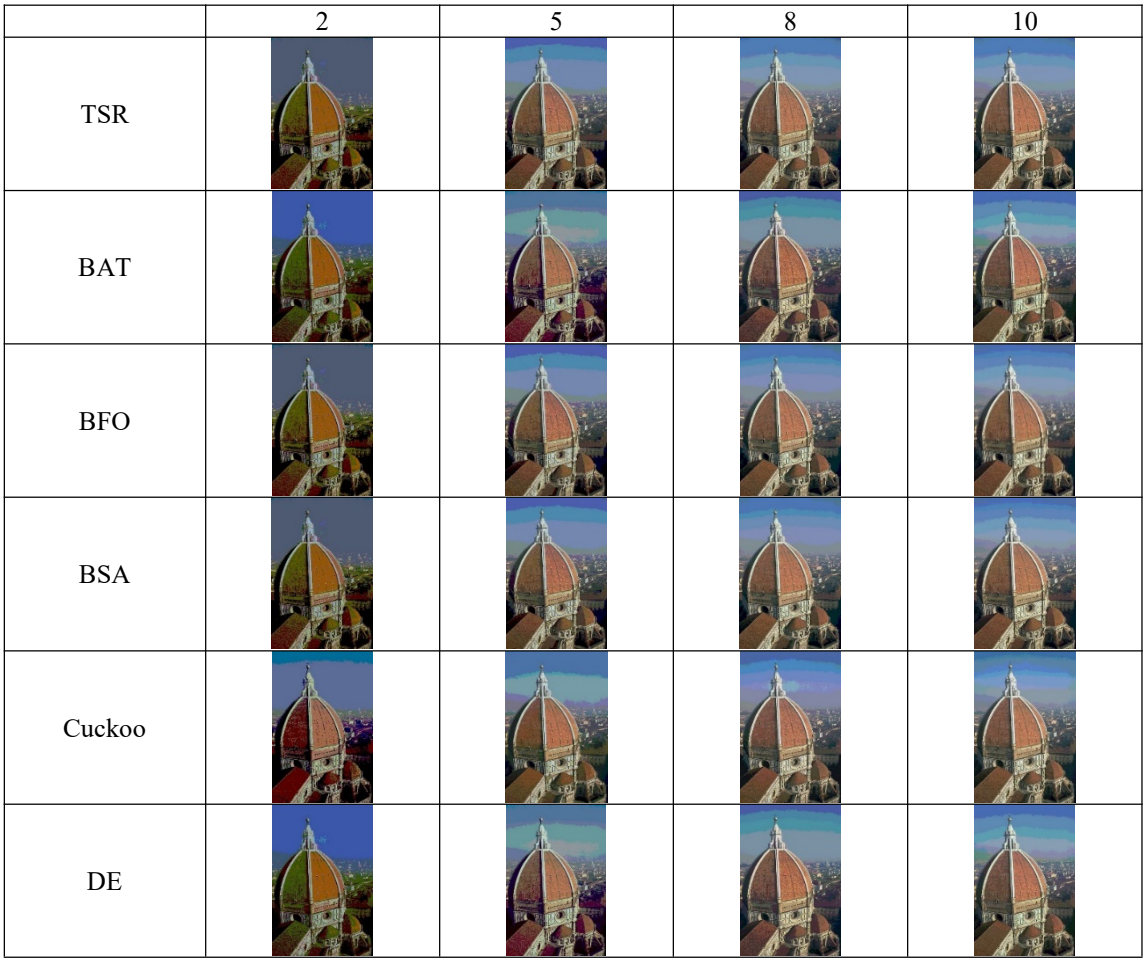


Figure 6. Otsu's -based multilevel thresholding on Image 1 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10}k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.



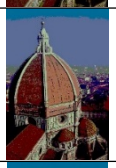
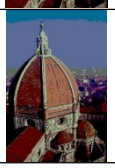
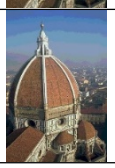
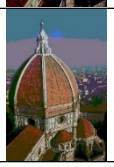
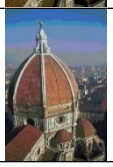
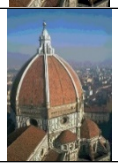
EFO				
FireFly				
PSO				
ABC				
WDO				

Figure 7. Kapur-based multilevel thresholding on Image 2 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10}k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.

	2	5	8	10
TSR				
BAT				
BFO				
BSA				
Cuckoo				

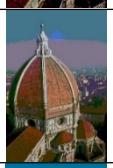
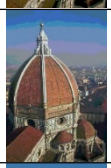
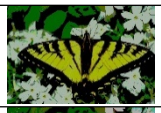
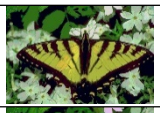
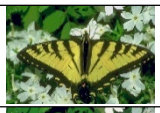
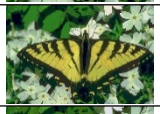
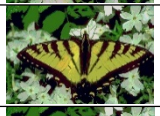
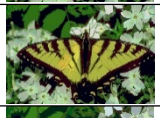
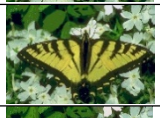
DE				
EFO				
FireFly				
PSO				
ABC				
WDO				

Figure 8. Otsu's -based multilevel thresholding on Image 2 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10}k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.

	2	5	8	10
TSR				
BAT				
BFO				
BSA				
Cuckoo				
DE				
EFO				

FireFly	
PSO	
ABC	
WDO	

Figure 9. Kapur-based multilevel thresholding on Image 3 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10}k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.

	2	5	8	10
TSR				
BAT				
BFO				
BSA				
Cuckoo				
DE				
EFO				
FireFly				
PSO				
ABC				
WDO				

Figure 10. Otsu's -based multilevel thresholding on Image 3 (BSDS). Rows: algorithms (TSR, BAT, BFO, BSA, Cuckoo, DE, EFO, FireFly, PSO, ABC, WDO). Columns: threshold levels k={2,5,8,10}k=\{2,5,8,10\}k={2,5,8,10}. Each sub-image shows mean PSNR (dB) and SSIM (0–1) over 50 runs. TSR better preserves boundaries and texture at moderate k.

5.3. Quality of results

In this section, we present the results of multilevel image thresholding using different metaheuristic algorithms. We use two entropy-based criteria to obtain the optimal thresholds: Kapur's entropy and Otsu's entropy. We conduct experiments on six images of different types and sizes. For each image, we find the optimal thresholds for different values of k, where k is the number of thresholds or the number of classes minus one. We vary k from 2 to 10 with a step size of 3. We evaluate the quality of the threshold images using the fitness value of the entropy function. The fitness value is the objective function that the metaheuristic algorithms try to optimize. A lower fitness value indicates a better result.

The results of the experiments are shown in Tables 1 to 3 Each table corresponds to one of the images. In each table, we report the fitness values of Kapur's entropy function and Otsu's entropy function for each algorithm and each value of k. From these tables, we can compare the performance of the algorithms and the effect of k on the fitness value.

We find that our proposed algorithm, TSR, has the lowest and the best fitness value for most of the images and values of k. TSR outperforms the other ten algorithms in terms of the entropy function. TSR also produces precise and consistent segmentation of the images into foreground and background regions, as shown in Figures 5 to 10. We also find that Kapur's entropy function has a lower fitness value than Otsu's entropy function for most of the images and values of k. The Kapur's entropy function produces better results than the Otsu's entropy function for this problem. Kapur's entropy function also has a lower execution time than Otsu's entropy function for most of the algorithms. We also find that the value of k affects the fitness value and the quality of the thresholding. Generally, as k increases, the fitness value increases, and the quality decreases. However, there are some exceptions where a higher value of k produces a lower fitness value or a better result. We also find that the complexity and the background of the image affect the performance of the algorithms. Generally, the more complex and darker the image is, the better TSR performs compared to the other algorithms. TSR can handle the noise and the contrast of the image well, while the other algorithms struggle to find the optimal thresholds.

The experimental results are summarized in Tables 1-3 which present Kapur's and Otsu's entropy functions for various algorithms and values of k. TSR consistently outperforms the other algorithms, achieving the lowest entropy values and demonstrating superior segmentation precision, as depicted in Figures 5-10. Kapur's entropy function generally provides better results than Otsu's, with lower execution times for most algorithms. The fitness value and segmentation quality typically improve with increasing k, though exceptions exist. TSR excels in complex and darker images, effectively managing noise and contrast.

Additional metrics Global Consistency Error (GCE), Probabilistic Rand Index (PRI), Variation of Information (VOI), Peak Signal-to-Noise Ratio (PSNR), Feature Similarity Index (FSIM), and Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE)—are detailed in Tables 4-9. TSR achieves the best scores across all metrics, indicating superior segmentation quality and consistency. Kapur's entropy function is preferable to Otsu's for this application. Future work should investigate TSR's performance on diverse images and consider enhancing its efficiency and robustness with adaptive or hybrid strategies.

Table 1. Fitness values of image 1

	K	Colour	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1 kapur	2	R	12.294578	12.2915684	12.265498	12.1545436	12.255878	12.2141545	12.215487	12.487890	12.221545	12.215488	12.165454
		G	11.448782	11.403050	11.403050	11.403050	11.403050	11.403050	11.403050	11.403050	11.403050	11.403050	11.402998
		B	11.548787	11.281663	11.281478	11.281663	11.281663	11.281663	11.281663	11.281663	11.281663	11.281663	11.281584
	5	R	21.031903	21.030366	21.025004	21.003236	21.034637	21.042328	20.878209	20.956416	21.034637	20.878209	20.816124
		G	19.751055	19.794187	19.796310	19.793889	19.797222	19.796574	19.796615	19.796748	19.797222	19.796615	19.626686
		B	18.729892	18.887606	18.888730	18.887239	18.878486	18.888530	18.775965	18.887909	18.878486	18.775965	18.700802
	8	R	28.055990	28.054417	28.058352	28.055487	28.061566	28.059131	28.061638	28.048979	28.061566	28.061638	27.760383
		G	27.001710	27.017876	27.003858	27.0155210	27.005196	27.0119786	27.021176	27.001105	27.005196	27.021176	26.356683
		B	26.092771	26.241450	26.143653	26.240878	26.174391	26.238050	26.227804	26.236016	26.174391	26.227804	25.695290
	10	R	32.057228	32.067446	32.062962	32.070388	32.060278	32.056516	32.074887	32.075519	32.060278	32.074887	31.739260
		G	31.057274	31.237465	30.952461	31.190502	31.032936	31.092619	31.103508	31.207569	31.032936	31.103508	30.400521
		B	30.315514	30.551967	30.304227	30.547003	30.290965	30.456497	30.454080	30.561148	30.290965	30.454080	29.741036
otsu	2	R	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8312	1848.8246
		G	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9308	1047.9100
		B	991.40433	991.40433	991.40433	991.40433	991.40433	991.40433	991.40433	991.40433	991.40433	991.40433	991.39895
	5	R	2107.6433	2107.6465	2107.6472	2106.1454	2107.6472	2105.1458	2107.6472	2107.6472	2107.6472	2107.6472	2099.5810
		G	1215.8271	1215.8289	1215.8289	1215.8289	1215.8289	1215.8275	1215.8289	1215.8289	1215.8289	1215.8289	1207.8897
		B	1209.6561	1209.6613	1208.9458	1209.5215	1209.14547	1206.1224	1201.8445	1200.1148	1199.1454	1208.1124	1198.9441
	8	R	2142.9055	2142.9742	2142.9467	2142.9795	2142.9696	2142.9503	2142.9703	2142.9788	2142.9696	2142.9703	2136.2455
		G	1245.7437	1245.7947	1245.6794	1245.7966	1245.7944	1245.7782	1245.6820	1245.7968	1245.7944	1245.6820	1237.4876
		B	1240.7066	1240.7664	1240.7666	1240.7754	1240.7624	1240.7457	1240.7478	1240.7754	1240.7624	1240.7664	1233.1521
	10	R	2151.5508	2151.7445	2151.5807	2151.7516	2151.6248	2150.6560	2148.6117	2149.7570	2151.6248	2151.7445	2146.3745
		G	1254.4264	1254.4488	1249.5314	1254.5626	1254.4166	1254.4525	1254.4607	1254.5588	1254.4166	1250.4488	1247.7149
		B	1248.6223	1243.6957	1245.7244	1248.8211	1248.6846	1248.5334	1248.4866	1248.8328	1248.6846	1246.6957	1243.0356

Table 2. Fitness values of image 2

	K	Colour	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 2 kapur	2	R	12.702121	12.702121	12.702121	12.702121	12.702121	12.702121	12.702121	12.702121	12.702121	12.702121	12.701896
		G	12.272001	12.272001	12.272001	12.272001	12.272001	12.272001	12.272001	12.272001	12.272001	12.272001	12.271890
		B	12.567272	12.567272	12.567272	12.567272	12.567272	12.567272	12.567272	12.567272	12.567272	12.567272	12.565750
	5	R	21.621497	21.621287	21.621418	21.621207	21.621534	21.621208	21.621484	21.621254	21.621534	21.621287	21.592733
		G	20.301826	20.300886	20.295287	20.30018	20.301862	20.300753	20.301807	20.3011196	20.301862	20.300886	20.225679
		B	21.101447	21.095670	21.095169	21.098768	21.101938	21.102155	21.102100	21.096283	21.101938	21.095670	21.027340
	8	R	29.048428	29.050683	29.046852	29.051759	29.051878	29.049261	29.049136	29.051064	29.051878	29.050683	28.885892
		G	27.018084	27.019959	27.018414	27.021286	27.023012	27.016488	27.019820	27.020738	27.023012	27.019959	26.824234
		B	28.176694	28.215559	28.175845	28.212806	28.178609	28.189916	28.182138	28.188811	28.178609	28.215559	28.105777
	10	R	33.406770	33.451707	33.384690	33.371454	33.403845	33.337967	33.340673	33.392079	33.403845	33.451707	32.922182
		G	30.706307	30.768322	30.693262	30.760078	30.709459	30.722779	30.687167	30.730489	30.75657	30.768442	30.392419
		B	32.406350	32.404275	32.382758	32.399444	32.414234	32.390566	32.412764	32.409662	35.414234	34.421512	32.404275
otsu	2	R	3154.5198	3154.5198	3154.5198	3154.5198	3154.5198	3154.5198	3154.5198	3154.5198	3154.6574	3154.4265	3154.4998
		G	1446.2514	1446.2514	1446.2514	1446.2514	1446.2514	1446.2514	1446.2514	1446.2514	1446.1244	1446.5657	1446.2348
		B	1497.7029	1497.7029	1497.7029	1497.7029	1497.7029	1497.7029	1497.7029	1497.7029	1497.7565	1497.56574	1497.7004
	5	R	3408.0065	3408.0065	3407.1548	3411.9544	3418.9487	3408.0047	3408.0043	3408.0065	3416.5657	3412.2657	3406.4441
		G	1579.6046	1579.6102	1579.6102	1579.6102	1579.6102	1579.6099	1579.6102	1579.6102	1578.2657	1572.2657	1569.6511
		B	1657.0532	1657.0037	1657.0264	1657.0048	1657.0588	1657.0169	1657.0203	1657.0242	1659.5467	1658.5575	1647.8564
	8	R	3456.3291	3456.4767	3456.3732	3456.4781	3456.3125	3456.4497	3456.4291	3456.4826	3478.6457	3562.6821	3452.6821
		G	1608.2817	1608.6150	1607.9583	1608.6161	1607.8449	1608.6039	1608.4153	1608.6171	1609.2347	1611.5657	1599.8342
		B	1701.7142	1701.9509	1701.7316	1701.9555	1701.5893	1701.9391	1701.8621	1701.9569	1697.5229	1698.5229	1694.5229
	10	R	3468.5617	3469.7650	3468.9095	3469.8057	3468.5095	3468.7043	3468.6594	3469.8060	3469.6132	3468.6132	3464.6132
		G	1626.5569	1616.5897	1612.5564	1616.5972	1616.0020	1616.4179	1616.3880	1616.5904	1617.5641	1612.9564	1610.9169
		B	1716.1214	1756.1456	1718.9546	179.6541	1713.4546	1711.8974	1714.2265	1710.4546	1718.4669	1721.0544	1706.0331

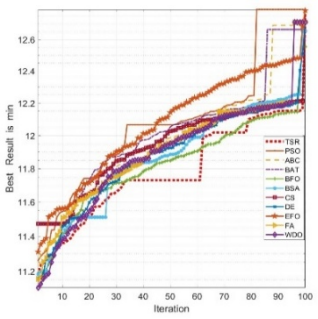
Table 3. Fitness values of image 3

	K	Colour	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 3 kapur	2	R	12.640256	12.658556	12.640256	12.791714	12.791714	12.365613	12.517071	12.639348	12.363913	12.366946	12.316912
		G	11.325326	11.868326	11.325326	11.173868	11.173868	11.254592	11.103134	11.254592	11.251508	11.251592	11.251540
		B	12.085576	11.785853	11.781663	11.933121	11.933121	11.433121	11.584579	11.433121	11.433121	11.433121	11.433042
	5	R	20.785836	20.725786	20.797336	20.948794	20.948794	21.193786	21.345244	21.107874	21.186095	21.029667	21.037582
		G	20.085533	20.189833	20.096733	20.248191	20.248191	19.948032	19.799490	19.948206	19.948680	19.948073	19.778144
		B	18.685836	18.606436	18.606436	18.757894	18.757894	18.740072	18.891530	18.739451	18.730044	18.927423	18.852260
	8	R	28.558779	28.412859	28.512779	28.664237	28.664237	28.210589	28.362047	28.200437	28.213024	28.213096	27.911841
		G	26.878593	27.129326	27.115308	27.536971	27.826646	27.833429	28.654879	28.648096	27.936758	27.442626	27.021176
		B	26.549899	26.352900	26.255103	26.762328	26.995841	26.059500	26.881050	27.817291	26.076553	26.649254	26.227804
	10	R	32.168678	32.178896	32.174412	32.591838	32.881728	32.877966	33.699416	33.703178	32.995862	32.496337	32.074887
		G	31.168724	31.348915	31.063911	31.711952	31.854386	31.914069	32.735519	32.675836	31.885361	31.524958	31.103508
		B	30.426964	30.663417	30.415677	31.068453	31.112415	31.277947	32.099397	31.933865	31.237127	30.875530	30.454080
otsu	2	R	1848.9426	1848.9426	1848.9426	1849.3526	1849.6526	1849.6526	1850.4740	1850.4740	1849.7640	1849.2526	1848.8312
		G	1048.0422	1048.0422	1048.0422	1048.4522	1048.7522	1048.7522	1049.5736	1049.5736	1048.8636	1048.3522	1047.9308
		B	991.51573	991.51573	991.51573	991.92573	992.22573	992.22573	993.04713	993.04713	992.33713	991.82573	991.40433
	5	R	2107.7547	2107.7579	2107.7586	2108.1686	2108.4686	2108.4677	2109.2891	2109.2900	2108.5800	2108.0686	2107.6472
		G	1215.9385	1215.9403	1215.9403	1216.3503	1216.6503	1216.6489	1217.4703	1217.4717	1216.7617	1216.2503	1218.8289
		B	1209.7675	1209.7727	1209.0572	1210.1833	1210.4833	1210.4818	1211.3032	1211.3047	1209.8786	1210.0819	1209.6605
	8	R	2143.0169	2143.0856	2143.0581	2143.5009	2143.791	2143.7717	2144.5931	2144.6124	2143.8795	2143.3917	2142.9703
		G	1245.8551	1245.9061	1245.7908	1246.318	1246.6158	1246.5996	1247.4210	1247.4372	1246.6122	1246.1034	1245.6820
		B	1240.818	1240.8778	1240.878	1241.2968	1241.5838	1241.5671	1242.3885	1242.4052	1241.6994	1241.1878	1240.7664
	10	R	2151.6622	2154.8559	2151.6921	2152.273	2152.4462	2152.4774	2155.3214	2153.2676	2152.5135	2152.1659	2151.7445
		G	1254.5378	1258.5602	1254.6428	1255.084	1255.238	1255.2739	1256.0953	1256.0594	1255.4642	1254.8702	1254.4488
		B	1249.7337	1252.8071	1248.8358	1249.3425	1249.506	1249.3548	1250.1762	1250.3274	1249.6572	1249.1171	1248.6957

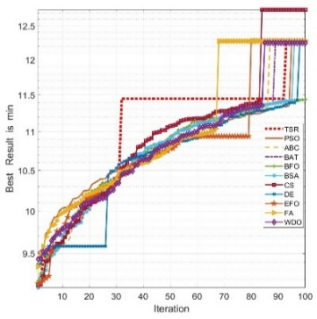
5.4. Plotting of results

This section presents the fitness value plots for Kapur's and Otsu's entropy functions across algorithms and images. Fitness values, which indicate the objective function optimized by the metaheuristic algorithms, are detailed in Tables 1-3. Figures 11 and 12 illustrate these values for example cases. TSR consistently achieves the lowest fitness values across most images and k values.

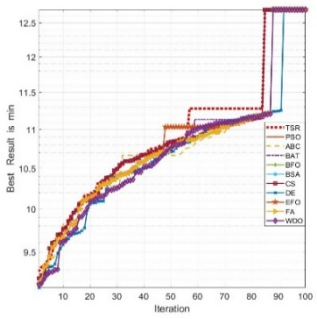
Figure 11 displays Kapur's entropy function for image 2 across all color channels (red, green, blue) and k values (2, 5, 8, 10), showing TSR's superior performance. Figure 12 shows Otsu's entropy function for the same image, where TSR also performs best in most channels and k values, except for a minor discrepancy in the blue channel with k=10, where PSO slightly outperforms TSR. These plots, derived from 100 iterations per algorithm, graphically represent the results reported in the tables, confirming TSR's superior performance in terms of fitness value for most images and k values.



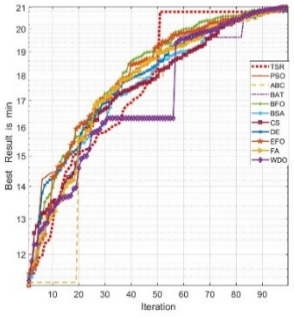
(a) K=2, Blue



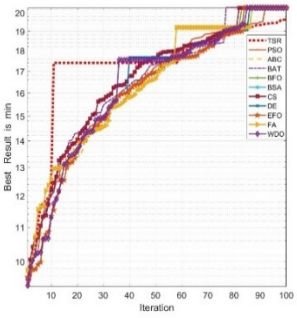
(b) K=2, Green



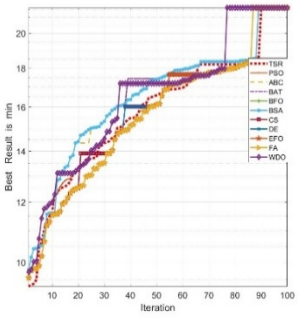
(c) K=2, Red



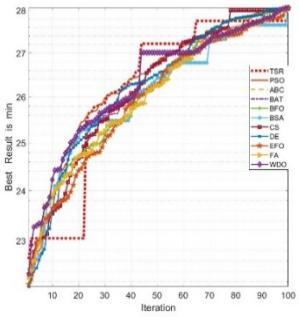
(d) K=5, Blue



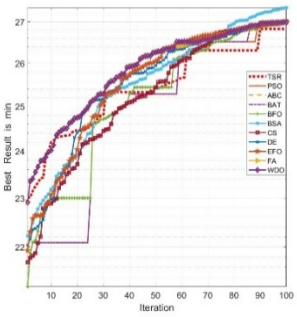
(e) K=5, Green



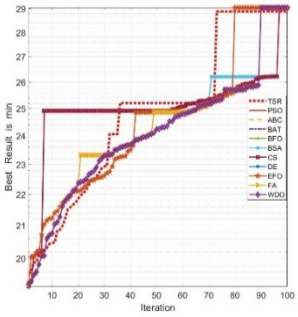
(f) K=5, Red



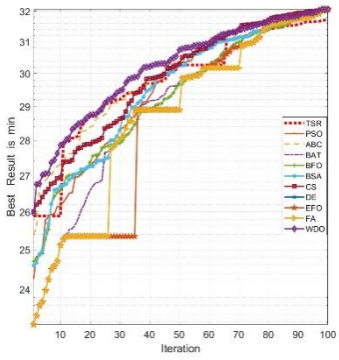
(g) K=8, Blue



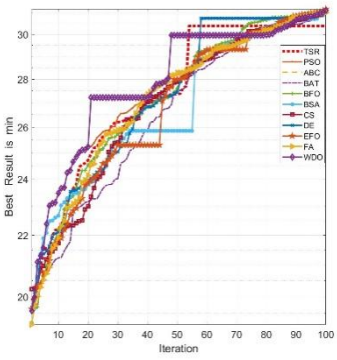
(h) K=8, Green



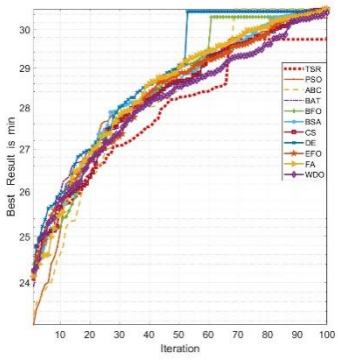
(i) K=8, Red



(j) K=10, Blue

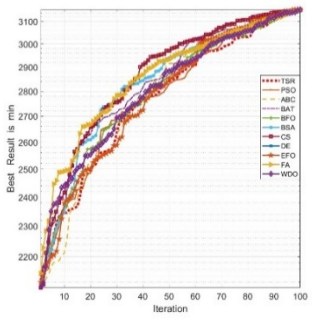


(k) K=10, Green

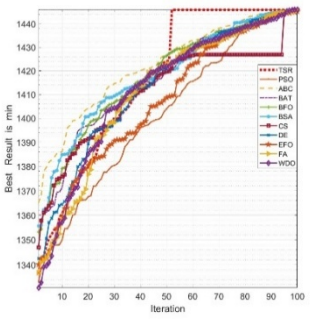


(l) K=10, Red

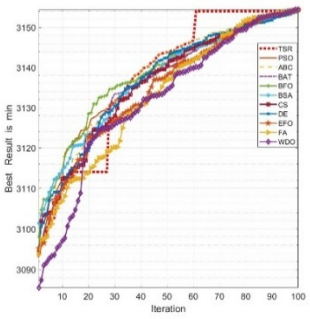
Figure 11. Converge curve solving Kapur's method for image 2



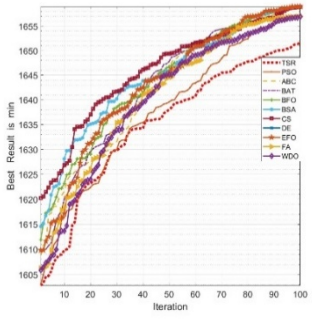
(a) K=2, Blue



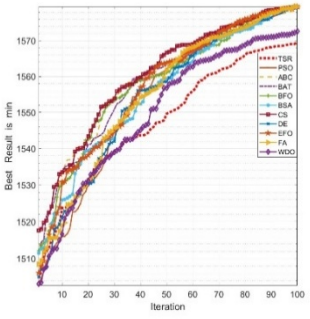
(b) K=2, Green



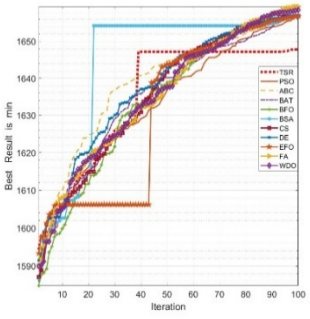
(c) K=2, Red



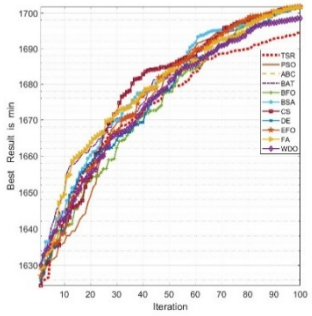
(d) K=5, Blue



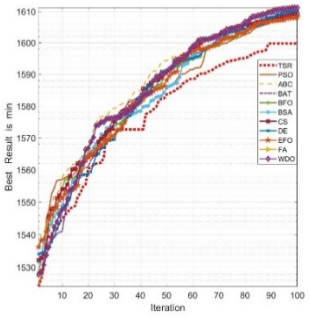
(e) K=5, Green



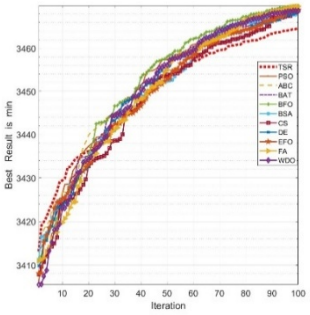
(f) K=5, Red



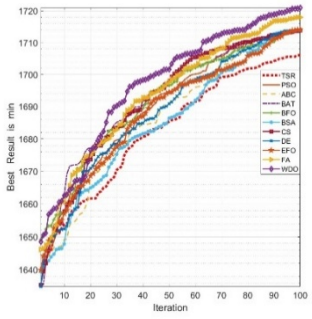
(g) K=8, Blue



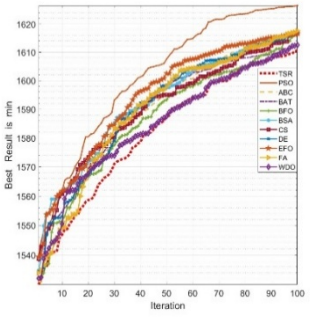
(h) K=8, Green



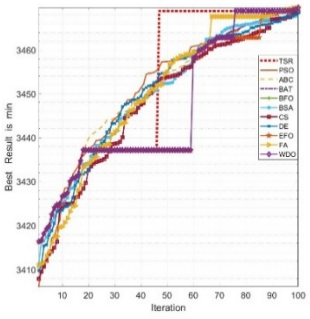
(i) K=8, Red



(j) K=10, Blue



(k) K=10, Green



(l) K=10, Red

Figure 12. Converge curve solving Otsu's method for image

Table 4. GCE values

		K	PSO	ABC	BAT	BFO	BSAb	CS	DE	EFO	FA	WDO	TSR
Img 1	Kapur	2	0.7401± 0.0011	0.7401± 0.0012	0.7401± 0.0010	0.7401± 0.0013	0.7401± 0.0011	0.7402± 0.0012	0.7411± 0.0014	0.7401± 0.0013	0.7411± 0.0011	0.7411± 0.0015	**0.7412± 0.0013**
		5	0.8090± 0.0016	0.8087± 0.0015	0.8121± 0.0018	0.8110± 0.0017	0.8095± 0.0014	0.8087± 0.0016	0.8115± 0.0018	0.8126± 0.0019	0.8124± 0.0015	0.8120± 0.0017	**0.8128± 0.0016**
		8	0.8375± 0.0019	0.8361± 0.0017	0.8377± 0.0016	0.8341± 0.0020	0.8401± 0.0018	0.8344± 0.0019	0.8341± 0.0017	0.8472± 0.0019	0.8471± 0.0018	0.8471± 0.0019	**0.8481± 0.0020**
		10	0.8277± 0.0016	0.8461± 0.0018	0.8346± 0.0017	0.8469± 0.0016	0.8383± 0.0019	0.8370± 0.0015	0.8424± 0.0017	0.8400± 0.0018	0.8409± 0.0017	0.8408± 0.0016	**0.8484± 0.0019**
	Otsu	2	0.7380± 0.0012	0.7380± 0.0011	0.7380± 0.0010	0.7380± 0.0012	0.7380± 0.0011	0.7381± 0.0013	0.7381± 0.0012	0.7380± 0.0011	0.7380± 0.0011	0.7386± 0.0012	**0.7379± 0.0012**
		5	0.8171± 0.0016	0.8152± 0.0017	0.8155± 0.0015	0.8132± 0.0016	0.8171± 0.0017	0.8126± 0.0018	0.8121± 0.0019	0.8151± 0.0017	0.8150± 0.0016	0.8141± 0.0018	**0.8167± 0.0017**
		8	0.8365± 0.0019	0.8395± 0.0017	0.8378± 0.0018	0.8405± 0.0019	0.8431± 0.0020	0.8368± 0.0019	0.8381± 0.0018	0.8393± 0.0019	0.8393± 0.0020	0.8393± 0.0018	**0.8394± 0.0019**
		10	0.8269± 0.0017	0.8255± 0.0018	0.8306± 0.0019	0.8267± 0.0018	0.8362± 0.0017	0.8223± 0.0019	0.8322± 0.0016	0.8322± 0.0018	0.8322± 0.0017	0.8322± 0.0019	**0.8235± 0.0018**
Img 2	Kapur	2	0.7068± 0.0011	0.7061± 0.0012	0.7057± 0.0013	0.7067± 0.0011	0.7066± 0.0012	0.7067± 0.0011	0.7066± 0.0012	0.7060± 0.0013	0.7066± 0.0012	0.7069± 0.0012	**0.7070± 0.0013**
		5	0.7694± 0.0015	0.7667± 0.0017	0.7700± 0.0016	0.7652± 0.0018	0.7684± 0.0015	0.7683± 0.0017	0.7670± 0.0016	0.7664± 0.0015	0.7660± 0.0018	0.7664± 0.0016	**0.7729± 0.0017**
		8	0.7847± 0.0018	0.7881± 0.0019	0.7858± 0.0020	0.7812± 0.0019	0.7841± 0.0020	0.7845± 0.0017	0.7875± 0.0019	0.7852± 0.0018	0.7852± 0.0019	0.7850± 0.0020	**0.7889± 0.0019**
		10	0.7862± 0.0019	0.7886± 0.0020	0.7893± 0.0018	0.7895± 0.0019	0.7879± 0.0019	0.7893± 0.0020	0.7933± 0.0018	0.7890± 0.0019	0.7861± 0.0019	0.7869± 0.0019	**0.7948± 0.0020**
	Otsu	2	0.7190± 0.0011	0.7190± 0.0012	0.7190± 0.0013	0.7189± 0.0011	0.7190± 0.0012	0.7190± 0.0011	0.7188± 0.0012	0.7180± 0.0013	0.7189± 0.0012	0.7185± 0.0011	**0.7190± 0.0013**
		5	0.7841± 0.0016	0.7849± 0.0015	0.7881± 0.0016	0.7872± 0.0017	0.7878± 0.0015	0.7844± 0.0017	0.7878± 0.0016	0.7881± 0.0017	0.7881± 0.0015	0.7889± 0.0018	**0.7798± 0.0017**
		8	0.7935± 0.0019	0.8028± 0.0020	0.7980± 0.0019	0.8044± 0.0018	0.7987± 0.0020	0.7983± 0.0019	0.7945± 0.0020	0.7954± 0.0019	0.7944± 0.0018	0.7944± 0.0019	**0.7936± 0.0020**
		10	0.7969± 0.0019	0.7955± 0.0018	0.7950± 0.0019	0.7962± 0.0020	0.7958± 0.0019	0.7921± 0.0020	0.7994± 0.0019	0.7963± 0.0020	0.7963± 0.0018	0.7942± 0.0019	**0.7985± 0.0020**
Img 3	Kapur	2	0.7072± 0.0011	0.7072± 0.0010	0.7071± 0.0011	0.7072± 0.0013	0.7071± 0.0012	0.7077± 0.0013	0.7071± 0.0012	0.7071± 0.0012	0.7074± 0.0011	0.7080± 0.0012	**0.7098± 0.0013**
		5	0.7170± 0.0015	0.7203± 0.0016	0.7271± 0.0017	0.7264± 0.0016	0.7255± 0.0017	0.7176± 0.0018	0.7281± 0.0016	0.7314± 0.0019	0.7307± 0.0017	0.7327± 0.0018	**0.7337± 0.0019**
		8	0.7593± 0.0019	0.7525± 0.0018	0.7515± 0.0019	0.7550± 0.0018	0.7571± 0.0020	0.7522± 0.0018	0.7487± 0.0019	0.7527± 0.0018	0.7512± 0.0019	0.7496± 0.0018	**0.7638± 0.0020**
		10	0.7511± 0.0017	0.7569± 0.0018	0.7561± 0.0019	0.7545± 0.0018	0.7531± 0.0019	0.7652± 0.0017	0.7628± 0.0019	0.7546± 0.0018	0.7537± 0.0019	0.7520± 0.0017	**0.7586± 0.0018**
	Otsu	2	0.7069± 0.0012	0.7069± 0.0013	0.7069± 0.0011	0.7069± 0.0012	0.7068± 0.0013	0.7069± 0.0012	0.7069± 0.0013	0.7041± 0.0012	0.7027± 0.0013	0.7014± 0.0012	**0.7070± 0.0013**
		5	0.7377± 0.0017	0.7419± 0.0018	0.7415± 0.0019	0.7417± 0.0018	0.7417± 0.0019	0.7415± 0.0017	0.7406± 0.0018	0.7405± 0.0019	0.7404± 0.0018	0.7410± 0.0017	**0.7453± 0.0019**
		8	0.7494± 0.0019	0.7522± 0.0018	0.7516± 0.0019	0.7519± 0.0018	0.7558± 0.0020	0.7491± 0.0019	0.7473± 0.0018	0.7515± 0.0019	0.7504± 0.0018	0.7515± 0.0019	**0.7524± 0.0020**
		10	0.7437± 0.0018	0.7373± 0.0019	0.7422± 0.0017	0.7385± 0.0018	0.7415± 0.0019	0.7468± 0.0018	0.7402± 0.0019	0.7370± 0.0017	0.7390± 0.0018	0.7388± 0.0019	**0.7505± 0.0020**

Table 5. PRI values

		K	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1	Kapur	2	0.9514± 0.0009	0.9514± 0.0008	0.9514± 0.0009	0.9514± 0.0008	0.9512± 0.0009	0.9511± 0.0008	0.9513± 0.0008	0.9513± 0.0009	0.9514± 0.0008	0.9514± 0.0009	**0.9515± 0.0008**
		5	0.9727± 0.0007	0.9727± 0.0008	0.9728± 0.0007	0.9725± 0.0009	0.9733± 0.0008	0.9731± 0.0008	0.9734± 0.0008	0.9732± 0.0007	0.9744± 0.0009	0.9733± 0.0008	**0.9744± 0.0008**
		8	0.9815± 0.0008	0.9815± 0.0009	0.9819± 0.0008	0.9812± 0.0009	0.9820± 0.0009	0.9815± 0.0008	0.9829± 0.0009	0.9814± 0.0008	0.9820± 0.0009	0.9820± 0.0008	**0.9829± 0.0008**
		10	0.9833± 0.0009	0.9839± 0.0008	0.9834± 0.0008	0.9834± 0.0009	0.9833± 0.0008	0.9837± 0.0008	0.9838± 0.0009	0.9841± 0.0008	0.9838± 0.0008	0.9833± 0.0009	**0.9841± 0.0008**
	Otsu	2	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9509± 0.0008	0.9516± 0.0008	**0.9516± 0.0008**
		5	0.9768± 0.0008	0.9764± 0.0009	0.9767± 0.0008	0.9761± 0.0008	0.9765± 0.0008	0.9760± 0.0009	0.9764± 0.0008	0.9764± 0.0009	0.9778± 0.0008	0.9765± 0.0009	**0.9778± 0.0008**
		8	0.9835± 0.0009	0.9835± 0.0008	0.9833± 0.0009	0.9834± 0.0008	0.9835± 0.0009	0.9836± 0.0008	0.9837± 0.0008	0.9836± 0.0009	0.9824± 0.0008	0.9835± 0.0009	**0.9836± 0.0008**
		10	0.9845± 0.0009	0.9849± 0.0008	0.9852± 0.0009	0.9843± 0.0008	0.9847± 0.0009	0.9848± 0.0008	0.9802± 0.0009	0.9848± 0.0008	0.9830± 0.0008	0.9847± 0.0009	**0.9848± 0.0008**
Image 2	Kapur	2	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9335± 0.0008	0.9292± 0.0008	0.9335± 0.0008	**0.9335± 0.0008**
		5	0.9712± 0.0008	0.9711± 0.0008	0.9711± 0.0009	0.9713± 0.0008	0.9724± 0.0008	0.9715± 0.0008	0.9714± 0.0008	0.9714± 0.0009	0.9734± 0.0008	0.9724± 0.0009	**0.9734± 0.0008**
		8	0.9817± 0.0009	0.9818± 0.0008	0.9817± 0.0008	0.9816± 0.0009	0.9818± 0.0008	0.9818± 0.0009	0.9817± 0.0008	0.9812± 0.0008	0.9792± 0.0009	0.9818± 0.0009	**0.9818± 0.0008**
		10	0.9836± 0.0008	0.9835± 0.0009	0.9838± 0.0008	0.9835± 0.0009	0.9839± 0.0008	0.9842± 0.0008	0.9836± 0.0009	0.9838± 0.0008	0.9824± 0.0008	0.9839± 0.0009	**0.9842± 0.0008**
	Otsu	2	0.9111± 0.0008	0.9111± 0.0008	0.9111± 0.0009	0.9111± 0.0008	0.9111± 0.0009	0.9111± 0.0008	0.9111± 0.0008	0.9111± 0.0008	0.9128± 0.0009	0.9111± 0.0009	**0.9128± 0.0008**
		5	0.9311± 0.0009	0.9320± 0.0008	0.9319± 0.0009	0.9293± 0.0008	0.9339± 0.0009	0.9332± 0.0008	0.9352± 0.0008	0.9370± 0.0009	0.9300± 0.0008	0.9339± 0.0009	**0.9370± 0.0008**
		8	0.9663± 0.0008	0.9650± 0.0009	0.9665± 0.0008	0.9636± 0.0009	0.9650± 0.0008	0.9632± 0.0008	0.9645± 0.0009	0.9648± 0.0008	0.9585± 0.0009	0.9650± 0.0009	**0.9665± 0.0008**
		10	0.9680± 0.0008	0.9683± 0.0009	0.9679± 0.0008	0.9692± 0.0009	0.9684± 0.0008	0.9686± 0.0009	0.9680± 0.0008	0.9688± 0.0009	0.9674± 0.0008	0.9684± 0.0009	**0.9692± 0.0008**
Image 3	Kapur	2	0.9514± 0.0009	0.9514± 0.0008	0.9514± 0.0009	0.9514± 0.0008	0.9512± 0.0009	0.9511± 0.0008	0.9513± 0.0008	0.9513± 0.0009	0.9514± 0.0008	0.9514± 0.0009	**0.9515± 0.0008**
		5	0.9727± 0.0007	0.9727± 0.0008	0.9728± 0.0007	0.9725± 0.0009	0.9733± 0.0008	0.9731± 0.0008	0.9734± 0.0008	0.9732± 0.0007	0.9744± 0.0009	0.9733± 0.0008	**0.9744± 0.0008**
		8	0.9815± 0.0008	0.9815± 0.0009	0.9819± 0.0008	0.9812± 0.0009	0.9820± 0.0009	0.9815± 0.0008	0.9829± 0.0009	0.9814± 0.0008	0.9820± 0.0009	0.9820± 0.0008	**0.9829± 0.0008**
		10	0.9833± 0.0009	0.9839± 0.0008	0.9834± 0.0008	0.9834± 0.0009	0.9833± 0.0008	0.9837± 0.0008	0.9838± 0.0009	0.9841± 0.0008	0.9838± 0.0008	0.9833± 0.0009	**0.9841± 0.0008**
	Otsu	2	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9516± 0.0008	0.9509± 0.0008	0.9516± 0.0008	**0.9516± 0.0008**
		5	0.9768± 0.0008	0.9764± 0.0009	0.9767± 0.0008	0.9761± 0.0008	0.9765± 0.0008	0.9760± 0.0009	0.9764± 0.0008	0.9764± 0.0009	0.9778± 0.0008	0.9765± 0.0009	**0.9778± 0.0008**
		8	0.9835± 0.0009	0.9835± 0.0008	0.9833± 0.0009	0.9834± 0.0008	0.9835± 0.0009	0.9836± 0.0008	0.9837± 0.0008	0.9836± 0.0009	0.9824± 0.0008	0.9835± 0.0009	**0.9836± 0.0008**
		10	0.9845± 0.0009	0.9849± 0.0008	0.9852± 0.0009	0.9843± 0.0008	0.9847± 0.0009	0.9848± 0.0008	0.9802± 0.0009	0.9848± 0.0008	0.9830± 0.0008	0.9847± 0.0009	**0.9848± 0.0008**

Table 6. VOI values

	K	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1	Kapur	2	7.7415± 0.0041	7.7414± 0.0040	7.7415± 0.0042	7.7315± 0.0041	7.6314± 0.0043	7.7404± 0.0042	7.7411± 0.0040	7.7403± 0.0043	7.7423± 0.0042	7.7404± 0.0041	**7.7432± 0.0042**
		5	7.6464± 0.0043	7.6387± 0.0044	7.6374± 0.0042	7.6564± 0.0043	7.6543± 0.0043	7.6191± 0.0045	7.6487± 0.0041	7.6543± 0.0042	7.6536± 0.0041	7.6517± 0.0042	**7.6899± 0.0041**
		8	7.3210± 0.0045	7.2686± 0.0042	7.2953± 0.0041	7.2793± 0.0042	7.2812± 0.0044	7.3214± 0.0043	7.3031± 0.0042	7.2791± 0.0043	7.2683± 0.0041	7.2682± 0.0041	**7.3099± 0.0042**
		10	7.0872± 0.0044	7.0398± 0.0041	7.1561± 0.0043	7.1228± 0.0041	7.1627± 0.0042	7.1029± 0.0043	7.0700± 0.0041	7.1521± 0.0043	7.1145± 0.0042	7.0394± 0.0041	**7.1964± 0.0042**
	Otsu	2	7.7454± 0.0041	7.7454± 0.0042	7.7454± 0.0042	7.7432± 0.0042	7.7454± 0.0042	7.7455± 0.0041	7.7454± 0.0042	7.7454± 0.0042	7.7412± 0.0043	7.7412± 0.0043	**7.7584± 0.0042**
		5	7.5359± 0.0043	7.5208± 0.0041	7.5208± 0.0043	7.5078± 0.0043	7.5196± 0.0043	7.5327± 0.0042	7.5114± 0.0043	7.5071± 0.0042	7.5200± 0.0042	7.5205± 0.0043	**7.5586± 0.0042**
		8	7.1566± 0.0042	7.1757± 0.0043	7.1990± 0.0042	7.1566± 0.0042	7.2163± 0.0043	7.1735± 0.0043	7.1705± 0.0041	7.1563± 0.0042	7.1755± 0.0041	7.1756± 0.0042	**7.2519± 0.0043**
		10	6.8641± 0.0043	6.8898± 0.0042	6.8314± 0.0041	6.9008± 0.0043	6.8419± 0.0042	6.9389± 0.0043	6.8012± 0.0041	6.9079± 0.0042	6.8837± 0.0043	6.8895± 0.0042	**7.0459± 0.0043**
Image 2	Kapur	2	7.8948± 0.0042	7.8949± 0.0043	7.8949± 0.0041	7.8949± 0.0042	7.8949± 0.0042	7.8949± 0.0042	7.8948± 0.0043	7.8947± 0.0043	7.8940± 0.0042	7.8947± 0.0041	**7.8821± 0.0043**
		5	7.3636± 0.0043	7.3434± 0.0042	7.3520± 0.0042	7.3506± 0.0043	7.3401± 0.0041	7.3621± 0.0043	7.3503± 0.0042	7.3597± 0.0042	7.3410± 0.0043	7.3414± 0.0042	**7.3875± 0.0042**
		8	6.8731± 0.0041	6.8811± 0.0042	6.9014± 0.0043	6.8928± 0.0041	6.8584± 0.0043	6.8710± 0.0042	6.8697± 0.0043	6.8266± 0.0042	6.8155± 0.0041	6.8454± 0.0043	**7.0090± 0.0042**
		10	6.6895± 0.0043	6.7076± 0.0041	6.7293± 0.0042	6.7448± 0.0041	6.7194± 0.0042	6.6865± 0.0043	6.7206± 0.0041	6.7448± 0.0043	6.7155± 0.0042	6.7072± 0.0043	**6.8161± 0.0041**
	Otsu	2	7.7596± 0.0041	7.7596± 0.0041	7.7596± 0.0042	7.7596± 0.0041	7.7596± 0.0043	7.7596± 0.0041	7.7596± 0.0042	7.7596± 0.0043	7.7596± 0.0041	7.7596± 0.0041	**7.7572± 0.0043**
		5	7.2478± 0.0042	7.2738± 0.0043	7.2557± 0.0041	7.2682± 0.0042	7.2713± 0.0043	7.2677± 0.0042	7.2605± 0.0043	7.2682± 0.0042	7.2738± 0.0041	7.2733± 0.0043	**7.2962± 0.0042**
		8	6.8827± 0.0043	6.9065± 0.0042	6.9289± 0.0043	6.8688± 0.0041	6.9353± 0.0042	6.8957± 0.0043	6.9001± 0.0042	6.8521± 0.0043	6.9165± 0.0042	6.9064± 0.0042	**6.9402± 0.0043**
		10	6.6563± 0.0041	6.6547± 0.0042	6.6494± 0.0043	6.6643± 0.0041	6.6482± 0.0043	6.6636± 0.0042	6.6204± 0.0043	6.6643± 0.0042	6.6517± 0.0041	6.6228± 0.0043	**6.7413± 0.0042**
Image 3	Kapur	2	7.2419± 0.0043	7.2419± 0.0043	7.2419± 0.0043	7.2419± 0.0043	7.2419± 0.0042	7.2419± 0.0042	7.2419± 0.0043	7.2418± 0.0041	7.2384± 0.0043	7.2419± 0.0042	**7.2496± 0.0043**
		5	6.8934± 0.0043	6.9469± 0.0042	6.9237± 0.0041	6.9256± 0.0042	6.8870± 0.0043	6.9619± 0.0042	6.9549± 0.0043	6.9252± 0.0042	6.9461± 0.0041	6.9468± 0.0043	**6.9818± 0.0042**
		8	6.4523± 0.0042	6.4336± 0.0043	6.4471± 0.0041	6.4547± 0.0043	6.4597± 0.0042	6.4348± 0.0042	6.4693± 0.0043	6.4547± 0.0041	6.4331± 0.0043	6.4335± 0.0043	**6.4652± 0.0042**
		10	6.2465± 0.0042	6.2404± 0.0043	6.2667± 0.0043	6.3083± 0.0042	6.2394± 0.0043	6.2480± 0.0042	6.3146± 0.0041	6.3080± 0.0043	6.2407± 0.0042	6.2321± 0.0043	**6.2919± 0.0042**
	Otsu	2	7.2721± 0.0041	7.2727± 0.0042	7.2727± 0.0043	7.2727± 0.0041	7.2727± 0.0042	7.2727± 0.0043	7.2149± 0.0043	7.2455± 0.0042	7.2227± 0.0041	7.2325± 0.0043	**7.2719± 0.0042**
		5	6.5188± 0.0043	6.5484± 0.0042	6.5476± 0.0043	6.5371± 0.0041	6.5452± 0.0042	6.5452± 0.0042	6.5445± 0.0043	6.5273± 0.0043	6.5484± 0.0042	6.5067± 0.0041	**6.6140± 0.0042**
		8	6.2166± 0.0042	6.2219± 0.0043	6.2430± 0.0043	6.1997± 0.0041	6.2455± 0.0043	6.2499± 0.0042	6.2217± 0.0042	6.1995± 0.0041	6.2237± 0.0043	6.2122± 0.0043	**6.2654± 0.0042**
		10	5.9134± 0.0042	5.9379± 0.0041	5.9558± 0.0043	5.8726± 0.0042	5.9560± 0.0043	5.9141± 0.0043	5.9383± 0.0042	5.8615± 0.0041	5.9133± 0.0042	5.9379± 0.0043	**6.0753± 0.0042**

Table 7. PSNR values

		K	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1	Kapur	2	28.4453± 0.0341	28.4526± 0.0340	28.4541± 0.0338	28.4602± 0.0342	28.4518± 0.0341	28.4507± 0.0343	28.4520± 0.0342	28.4657± 0.0341	28.4749± 0.0343	28.4338± 0.0342	**28.4884± 0.0340**
		5	32.6648± 0.0284	32.6421± 0.0285	32.6369± 0.0283	32.6258± 0.0286	32.6637± 0.0285	32.6619± 0.0284	32.6305± 0.0285	32.6312± 0.0284	32.6607± 0.0286	32.6601± 0.0283	**32.6527± 0.0285**
		8	35.7319± 0.0247	35.7223± 0.0249	35.7186± 0.0248	35.7678± 0.0246	35.7291± 0.0248	35.7189± 0.0247	35.7812± 0.0245	35.4217± 0.0249	35.4326± 0.0248	35.4198± 0.0249	**35.7357± 0.0247**
		10	37.1384± 0.0238	37.1910± 0.0237	37.2305± 0.0236	37.1422± 0.0238	37.1448± 0.0237	37.2404± 0.0239	37.1135± 0.0238	37.1384± 0.0238	37.1910± 0.0237	37.2305± 0.0236	**37.2431± 0.0237**
	Otsu	2	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	28.1327± 0.0365	**28.1213± 0.0367**
		5	33.3468± 0.0292	33.3287± 0.0293	33.3264± 0.0294	33.3251± 0.0292	33.3375± 0.0294	33.3250± 0.0293	33.3294± 0.0294	33.3448± 0.0292	33.3287± 0.0293	33.3260± 0.0292	**33.2471± 0.0295**
		8	36.7175± 0.0251	36.6676± 0.0250	36.6847± 0.0252	36.6581± 0.0251	36.7333± 0.0253	36.6921± 0.0250	36.7187± 0.0252	36.7175± 0.0251	36.6676± 0.0250	36.6815± 0.0253	**36.7668± 0.0252**
		10	38.1618± 0.0229	38.2906± 0.0228	38.1630± 0.0229	38.3153± 0.0228	38.2145± 0.0229	38.1602± 0.0228	38.2232± 0.0229	38.1618± 0.0229	38.2906± 0.0228	38.1630± 0.0229	**38.5195± 0.0227**
Image 2	Kapur	2	25.8675± 0.0371	25.8679± 0.0372	25.8526± 0.0373	25.8355± 0.0371	25.8421± 0.0370	25.8314± 0.0372	25.8755± 0.0373	25.8357± 0.0370	25.8488± 0.0371	25.8374± 0.0372	**25.8843± 0.0371**
		5	31.8392± 0.0302	31.9233± 0.0303	32.0185± 0.0301	32.0046± 0.0304	31.8448± 0.0303	31.9383± 0.0302	31.9299± 0.0304	31.8246± 0.0303	31.9243± 0.0302	32.0385± 0.0301	**32.1901± 0.0302**
		8	36.2428± 0.0260	36.1435± 0.0262	36.3121± 0.0261	36.1963± 0.0260	36.2347± 0.0261	36.2339± 0.0260	36.1836± 0.0263	36.2428± 0.0261	36.1435± 0.0260	36.3121± 0.0262	**36.5294± 0.0260**
		10	37.3585± 0.0239	37.5412± 0.0238	37.5419± 0.0239	37.5442± 0.0237	37.3426± 0.0238	37.5464± 0.0239	37.5315± 0.0237	37.3596± 0.0238	37.5411± 0.0239	37.5401± 0.0238	**37.7521± 0.0237**
	Otsu	2	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5287± 0.0358	26.5202± 0.0359	**26.5456± 0.0358**
		5	33.0162± 0.0296	33.0114± 0.0297	33.0189± 0.0296	33.0131± 0.0297	33.0353± 0.0298	33.0123± 0.0296	33.0232± 0.0297	33.0162± 0.0296	33.0114± 0.0297	33.0189± 0.0296	**33.0210± 0.0297**
		8	36.1912± 0.0262	36.2367± 0.0261	36.1827± 0.0262	36.1783± 0.0263	36.3015± 0.0261	36.1853± 0.0263	36.1438± 0.0262	36.1901± 0.0263	36.2232± 0.0262	36.1812± 0.0263	**36.2067± 0.0261**
		10	38.0092± 0.0238	37.9887± 0.0239	37.9987± 0.0237	37.9901± 0.0238	37.9209± 0.0238	38.0464± 0.0239	38.0633± 0.0237	37.9765± 0.0238	37.9887± 0.0239	37.9765± 0.0238	**38.0115± 0.0238**
Image 3	Kapur	2	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2486± 0.0352	27.2484± 0.0352	27.2315± 0.0353	**27.2813± 0.0352**
		5	32.2103± 0.0300	32.0910± 0.0301	32.1372± 0.0299	32.4944± 0.0300	31.9430± 0.0301	31.8289± 0.0299	32.1697± 0.0300	32.2108± 0.0301	32.0934± 0.0300	32.1255± 0.0301	**31.9174± 0.0300**
		8	36.9239± 0.0260	36.7604± 0.0261	36.8585± 0.0262	36.7119± 0.0261	36.9763± 0.0260	36.5254± 0.0263	36.5589± 0.0261	36.9234± 0.0260	36.7602± 0.0261	36.8581± 0.0262	**36.9726± 0.0260**
		10	37.9645± 0.0239	37.8476± 0.0240	38.0639± 0.0238	38.0199± 0.0239	37.9524± 0.0240	37.8891± 0.0238	37.9011± 0.0239	37.9645± 0.0238	37.8416± 0.0240	38.0633± 0.0239	**38.1319± 0.0238**
	Otsu	2	28.6112± 0.0341	28.6123± 0.0340	28.6174± 0.0341	28.6123± 0.0340	28.6165± 0.0341	28.6301± 0.0342	28.6255± 0.0341	28.6186± 0.0342	28.6190± 0.0341	28.6362± 0.0340	**28.6335± 0.0341**
		5	35.2110± 0.0285	35.1902± 0.0284	35.1813± 0.0286	35.1872± 0.0285	35.1872± 0.0284	35.2005± 0.0286	35.1975± 0.0285	35.2109± 0.0284	35.1901± 0.0285	35.1811± 0.0286	**35.2504± 0.0284**
		8	37.6142± 0.0251	37.6063± 0.0252	37.6149± 0.0251	37.5794± 0.0252	37.5979± 0.0251	37.6079± 0.0252	37.6111± 0.0251	37.6142± 0.0252	37.5249± 0.0251	37.6149± 0.0252	**37.7360± 0.0250**
		10	38.9856± 0.0228	38.9752± 0.0229	39.0692± 0.0227	38.9599± 0.0228	38.9920± 0.0229	38.9466± 0.0228	39.1198± 0.0227	38.9846± 0.0229	38.9701± 0.0228	39.0579± 0.0228	**39.1255± 0.0227**

Table 8. FSIM values

		K	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1	Kapur	2	0.840662± 0.0004	0.840533± 0.0005	0.840552± 0.0005	0.840562± 0.0004	0.840514± 0.0004	0.840792± 0.0005	0.840005± 0.0005	0.840573± 0.0005	0.840197± 0.0004	0.840562± 0.0005	**0.841002± 0.0004**
		5	0.920813± 0.0007	0.920492± 0.0007	0.921743± 0.0007	0.921509± 0.0007	0.920805± 0.0007	0.920469± 0.0007	0.921499± 0.0007	0.921895± 0.0007	0.920268± 0.0007	0.920492± 0.0007	**0.922875± 0.0007**
		8	0.956183± 0.0006	0.956614± 0.0006	0.958268± 0.0006	0.957142± 0.0006	0.957229± 0.0006	0.956516± 0.0006	0.957197± 0.0006	0.957971± 0.0006	0.956188± 0.0006	0.956614± 0.0006	**0.957402± 0.0006**
		10	0.967606± 0.0005	0.967900± 0.0005	0.968166± 0.0005	0.967940± 0.0005	0.967691± 0.0005	0.968182± 0.0005	0.967790± 0.0005	0.968333± 0.0005	0.967604± 0.0005	0.967900± 0.0005	**0.969485± 0.0005**
	Otsu	2	0.842048± 0.0005	0.842684± 0.0005	0.842250± 0.0005	0.842290± 0.0005	0.842529± 0.0005	0.842143± 0.0005	0.842224± 0.0005	0.842527± 0.0005	0.842614± 0.0005	0.842232± 0.0005	**0.842783± 0.0005**
		5	0.930401± 0.0006	0.930385± 0.0006	0.930552± 0.0006	0.930139± 0.0006	0.930101± 0.0006	0.930459± 0.0006	0.930304± 0.0006	0.930381± 0.0006	0.930317± 0.0006	0.930587± 0.0006	**0.935242± 0.0006**
		8	0.961799± 0.0005	0.961618± 0.0005	0.961599± 0.0005	0.961444± 0.0005	0.961577± 0.0005	0.961681± 0.0005	0.961914± 0.0005	0.962505± 0.0005	0.961719± 0.0005	0.961718± 0.0005	**0.984125± 0.0005**
		10	0.972133± 0.0004	0.971577± 0.0004	0.970017± 0.0004	0.972097± 0.0004	0.972066± 0.0004	0.971792± 0.0004	0.972208± 0.0004	0.972214± 0.0004	0.972748± 0.0004	0.972003± 0.0004	**0.972302± 0.0004**
Image 2	Kapur	2	0.771302± 0.0011	0.770581± 0.0011	0.771482± 0.0011	0.773239± 0.0011	0.776570± 0.0011	0.777453± 0.0011	0.789525± 0.0011	0.754258± 0.0011	0.774418± 0.0011	0.775667± 0.0011	**0.796317± 0.0011**
		5	0.857914± 0.0009	0.858563± 0.0009	0.860280± 0.0009	0.859597± 0.0009	0.857932± 0.0009	0.858514± 0.0009	0.858850± 0.0009	0.859450± 0.0009	0.857907± 0.0009	0.858557± 0.0009	**0.863360± 0.0009**
		8	0.914325± 0.0008	0.914353± 0.0008	0.916420± 0.0008	0.914269± 0.0008	0.914219± 0.0008	0.914369± 0.0008	0.915254± 0.0008	0.914791± 0.0008	0.914325± 0.0008	0.914353± 0.0008	**0.916760± 0.0008**
		10	0.931497± 0.0006	0.931612± 0.0006	0.932383± 0.0006	0.931396± 0.0006	0.931273± 0.0006	0.931316± 0.0006	0.931203± 0.0006	0.931359± 0.0006	0.931496± 0.0006	0.931612± 0.0006	**0.932633± 0.0006**
	Otsu	2	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	0.774895± 0.0010	**0.774770± 0.0010**
		5	0.874580± 0.0008	0.874797± 0.0008	0.874765± 0.0008	0.874812± 0.0008	0.874546± 0.0008	0.874782± 0.0008	0.874727± 0.0008	0.874765± 0.0008	0.874575± 0.0008	0.874797± 0.0008	**0.875930± 0.0008**
		8	0.914917± 0.0007	0.915008± 0.0007	0.914993± 0.0007	0.914880± 0.0007	0.916860± 0.0007	0.915035± 0.0007	0.914744± 0.0007	0.915430± 0.0007	0.914907± 0.0007	0.915008± 0.0007	**0.916926± 0.0007**
		10	0.935916± 0.0005	0.933422± 0.0005	0.934926± 0.0005	0.933251± 0.0005	0.935796± 0.0005	0.936361± 0.0005	0.935604± 0.0005	0.933557± 0.0005	0.935916± 0.0005	0.933422± 0.0005	**0.948874± 0.0005**
Image 3	Kapur	2	0.822948± 0.0006	0.822980± 0.0006	0.822926± 0.0006	0.822981± 0.0006	0.822004± 0.0006	0.822860± 0.0006	0.822238± 0.0006	0.822014± 0.0006	0.822788± 0.0006	0.822514± 0.0006	**0.823004± 0.0006**
		5	0.896476± 0.0007	0.895172± 0.0007	0.895502± 0.0007	0.899033± 0.0007	0.894084± 0.0007	0.892961± 0.0007	0.897967± 0.0007	0.897031± 0.0007	0.896476± 0.0007	0.895172± 0.0007	**0.894258± 0.0007**
		8	0.944718± 0.0006	0.943194± 0.0006	0.944143± 0.0006	0.942884± 0.0006	0.945536± 0.0006	0.940671± 0.0006	0.940969± 0.0006	0.943041± 0.0006	0.944059± 0.0006	0.943194± 0.0006	**0.945209± 0.0006**
		10	0.954402± 0.0005	0.953654± 0.0005	0.953882± 0.0005	0.954450± 0.0005	0.954417± 0.0005	0.953768± 0.0005	0.953446± 0.0005	0.952821± 0.0005	0.954402± 0.0005	0.953654± 0.0005	**0.960001± 0.0005**
	Otsu	2	0.824565± 0.0007	0.826103± 0.0007	0.826259± 0.0007	0.825257± 0.0007	0.825247± 0.0007	0.824992± 0.0007	0.824128± 0.0007	0.824990± 0.0007	0.824568± 0.0007	0.825249± 0.0007	**0.826308± 0.0007**
		5	0.926223± 0.0005	0.925757± 0.0005	0.925609± 0.0005	0.925767± 0.0005	0.925767± 0.0005	0.925784± 0.0005	0.925966± 0.0005	0.925767± 0.0005	0.926223± 0.0005	0.925757± 0.0005	**0.924660± 0.0005**
		8	0.950126± 0.0005	0.949265± 0.0005	0.949468± 0.0005	0.949073± 0.0005	0.949582± 0.0005	0.949733± 0.0005	0.949625± 0.0005	0.949013± 0.0005	0.950130± 0.0005	0.949214± 0.0005	**0.949692± 0.0005**
		10	0.961431± 0.0004	0.961407± 0.0004	0.961162± 0.0004	0.961274± 0.0004	0.961955± 0.0004	0.960787± 0.0004	0.962239± 0.0004	0.961477± 0.0004	0.961431± 0.0004	0.961407± 0.0004	**0.961808± 0.0004**

Table 9. BRISQUE values

		K	PSO	ABC	BAT	BFO	BSA	CS	DE	EFO	FA	WDO	TSR
Image 1	Kapur	2	53.621± 0.010	53.621± 0.011	53.621± 0.011	53.621± 0.011	53.621± 0.010	53.621± 0.011	53.621± 0.011	53.621± 0.010	53.621± 0.011	53.621± 0.010	53.622± 0.010
		5	48.399± 0.040	48.039± 0.041	48.228± 0.041	46.228± 0.041	48.108± 0.041	48.108± 0.041	48.228± 0.041	48.227± 0.041	48.399± 0.041	48.039± 0.041	47.354± 0.041
		8	46.766± 0.042	47.003± 0.042	45.715± 0.042	47.003± 0.042	45.496± 0.042	46.505± 0.042	46.701± 0.042	45.094± 0.042	46.766± 0.042	47.003± 0.042	46.810± 0.042
		10	43.852± 0.045	44.925± 0.045	43.064± 0.045	45.114± 0.045	45.239± 0.045	44.554± 0.045	45.638± 0.045	44.919± 0.045	43.852± 0.045	44.925± 0.045	45.428± 0.045
	Otsu	2	53.090± 0.010	53.090± 0.011	53.090± 0.011	53.090± 0.011	53.090± 0.010	53.090± 0.011	53.090± 0.011	53.090± 0.010	53.090± 0.011	53.090± 0.010	53.090± 0.010
		5	47.682± 0.039	47.767± 0.039	47.749± 0.039	47.749± 0.039	47.767± 0.039	47.744± 0.039	47.749± 0.039	47.749± 0.039	47.682± 0.039	47.767± 0.039	46.745± 0.039
		8	44.413± 0.038	46.025± 0.038	46.188± 0.038	46.184± 0.038	44.963± 0.038	44.506± 0.038	44.210± 0.038	44.306± 0.038	44.413± 0.038	46.025± 0.038	46.722± 0.038
		10	42.656± 0.036	42.699± 0.036	42.757± 0.036	42.606± 0.036	42.410± 0.036	42.781± 0.036	42.502± 0.036	42.656± 0.036	42.656± 0.036	42.699± 0.036	43.548± 0.036
Image 2	Kapur	2	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.745± 0.011	53.744± 0.011	53.364± 0.011
		5	49.376± 0.042	49.350± 0.042	47.474± 0.042	49.376± 0.042	49.376± 0.042	49.885± 0.042	49.376± 0.042	49.592± 0.042	49.376± 0.042	49.350± 0.042	50.005± 0.042
		8	44.638± 0.040	44.489± 0.040	43.982± 0.040	44.599± 0.040	44.238± 0.040	44.544± 0.040	44.542± 0.040	43.850± 0.040	44.638± 0.040	44.489± 0.040	44.188± 0.040
		10	42.937± 0.039	43.102± 0.039	42.621± 0.039	41.916± 0.039	41.932± 0.039	42.923± 0.039	44.068± 0.039	42.707± 0.039	42.937± 0.039	43.102± 0.039	43.421± 0.039
	Otsu	2	52.976± 0.010	52.976± 0.011	52.976± 0.011	52.976± 0.011	52.976± 0.010	52.976± 0.011	52.976± 0.011	52.976± 0.010	52.976± 0.011	52.976± 0.010	52.976± 0.010
		5	46.225± 0.038	46.238± 0.038	46.281± 0.038	46.281± 0.038	46.281± 0.038	46.225± 0.038	46.281± 0.038	46.524± 0.038	46.225± 0.038	46.238± 0.038	45.096± 0.038
		8	43.563± 0.037	43.952± 0.037	43.933± 0.037	43.852± 0.037	42.644± 0.037	43.772± 0.037	43.120± 0.037	43.994± 0.037	43.563± 0.037	43.952± 0.037	43.653± 0.037
		10	41.945± 0.035	42.342± 0.035	42.967± 0.035	42.472± 0.035	42.642± 0.035	41.677± 0.035	43.943± 0.035	42.600± 0.035	41.945± 0.035	42.342± 0.035	43.333± 0.035
Image 3	Kapur	2	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012	49.115± 0.012
		5	45.142± 0.044	43.662± 0.044	48.543± 0.044	43.239± 0.044	43.654± 0.044	43.583± 0.044	48.335± 0.044	43.654± 0.044	45.142± 0.044	43.662± 0.044	49.220± 0.044
		8	48.118± 0.043	47.625± 0.043	48.415± 0.043	47.306± 0.043	47.965± 0.043	49.345± 0.043	48.116± 0.043	48.064± 0.043	48.118± 0.043	47.625± 0.043	47.266± 0.043
		10	47.750± 0.041	47.354± 0.041	47.374± 0.041	47.746± 0.041	47.324± 0.041	47.907± 0.041	48.046± 0.041	47.464± 0.041	47.750± 0.041	47.354± 0.041	48.618± 0.041
	Otsu	2	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.593± 0.010	51.561± 0.010
		5	49.999± 0.041	50.050± 0.041	50.091± 0.041	50.091± 0.041	50.091± 0.041	50.091± 0.041	50.091± 0.041	50.091± 0.041	49.999± 0.041	50.050± 0.041	49.756± 0.041
		8	48.768± 0.040	48.542± 0.040	48.652± 0.040	48.455± 0.040	48.731± 0.040	48.941± 0.040	48.412± 0.040	48.455± 0.040	48.768± 0.040	48.542± 0.040	48.914± 0.040
		10	44.971± 0.039	46.492± 0.039	46.349± 0.039	46.105± 0.039	45.683± 0.039	46.383± 0.039	44.959± 0.039	45.918± 0.039	44.971± 0.039	46.492± 0.039	44.917± 0.039
Image 4	Kapur	2	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011	51.815± 0.011
		5	45.359± 0.040	45.580± 0.040	45.438± 0.040	45.326± 0.040	45.489± 0.040	45.489± 0.040	45.489± 0.040	45.425± 0.040	45.359± 0.040	45.580± 0.040	46.371± 0.040
		8	42.880± 0.038	43.895± 0.038	43.090± 0.038	43.496± 0.038	42.701± 0.038	42.677± 0.038	42.698± 0.038	43.677± 0.038	42.880± 0.038	43.895± 0.038	42.816± 0.038
		10	42.026± 0.036	41.917± 0.036	41.933± 0.036	41.850± 0.036	42.462± 0.036	41.953± 0.036	42.010± 0.036	41.958± 0.036	42.026± 0.036	41.917± 0.036	42.050± 0.036
	Otsu	2	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.994± 0.010	46.830± 0.010
		5	45.377± 0.038	45.308± 0.038	45.308± 0.038	45.308± 0.038	45.377± 0.038	45.262± 0.038	45.377± 0.038	45.308± 0.038	45.377± 0.038	45.308± 0.038	44.813± 0.038
		8	42.303± 0.037	42.468± 0.037	42.304± 0.037	42.468± 0.037	42.043± 0.037	42.286± 0.037	42.297± 0.037	42.468± 0.037	42.303± 0.037	42.468± 0.037	42.630± 0.037
		10	41.619± 0.035	41.798± 0.035	41.461± 0.035	41.810± 0.035	42.307± 0.035	42.067± 0.035	42.309± 0.035	41.202± 0.035	41.619± 0.035	41.798± 0.035	41.650± 0.035
Image 5	Kapur	2	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.621± 0.010	53.461± 0.010
		5	48.399± 0.041	48.039± 0.041	48.228± 0.041	48.228± 0.041	48.108± 0.041	48.108± 0.041	48.228± 0.041	48.228± 0.041	48.399± 0.041	48.039± 0.041	47.354± 0.041
		8	46.766± 0.042	47.003± 0.042	45.715± 0.042	47.003± 0.042	45.496± 0.042	46.505± 0.042	46.701± 0.042	45.094± 0.042	46.766± 0.042	47.003± 0.042	46.810± 0.042
		10	43.852± 0.045	44.925± 0.045	43.064± 0.045	45.114± 0.045	45.239± 0.045	44.554± 0.045	45.638± 0.045	44.919± 0.045	43.852± 0.045	44.925± 0.045	42.423± 0.045
	Otsu	2	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010	53.090± 0.010
		5	47.682± 0.039	47.767± 0.039	47.749± 0.039	47.749± 0.039	47.767± 0.039	47.744± 0.039	47.749± 0.039	47.749± 0.039	47.682± 0.039	47.767± 0.039	46.745± 0.039
		8	44.413± 0.038	46.025± 0.038	46.188± 0.038	46.184± 0.038	44.963± 0.038	44.506± 0.038	44.210± 0.038	44.306± 0.038	44.413± 0.038	46.025± 0.038	46.722± 0.038
		10	42.656± 0.036	42.699± 0.036	42.757± 0.036	42.606± 0.036	42.410± 0.036	42.781± 0.036	42.502± 0.036	42.656± 0.036	42.656± 0.036	42.699± 0.036	43.548± 0.036
Image 6	Kapur	2	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.744± 0.011	53.745± 0.011	53.744± 0.011	53.364± 0.011
		5	49.376± 0.042	49.350± 0.042	47.474± 0.042	49.376± 0.042	49.376± 0.042	49.885± 0.042	49.376± 0.042	49.592± 0.042	49.376± 0.042	49.350± 0.042	50.005± 0.042
		8	44.638± 0.040	44.489± 0.040	43.982± 0.040	44.599± 0.040	44.238± 0.040	44.544± 0.040	44.542± 0.040	43.850± 0.040	44.638± 0.040	44.489± 0.040	44.188± 0.040
		10	42.937± 0.039	43.102± 0.039	42.621± 0.039	41.916± 0.039	41.932± 0.039	42.923± 0.039	44.068± 0.039	42.707± 0.039	42.937± 0.039	43.102± 0.039	43.421± 0.039
	Otsu	2	53.621± 0.010	53.621± 0.011	53.621± 0.011	53.621± 0.011	53.621± 0.010	53.621± 0.011	53.621± 0.011	53.621± 0.010	53.621± 0.011	53.621± 0.010	53.622± 0.010
		5	48.399± 0.040	48.039± 0.041	48.228± 0.041	46.228± 0.041	48.108± 0.041	48.108± 0.041	48.228± 0.041	48.227± 0.041	48.399± 0.041	48.039± 0.041	47.354± 0.041
		8	46.766± 0.042	47.003± 0.042	45.715± 0.042	47.003± 0.042	45.496± 0.042	46.505± 0.042	46.701± 0.042	45.094± 0.042	46.766± 0.042	47.003± 0.042	46.810± 0.042
		10	43.852± 0.045	44.925± 0.045	43.064± 0.045	45.114± 0.045	45.239± 0.045	44.554± 0.045	45.638± 0.045	44.919± 0.045	43.852± 0.045	44.925± 0.045	45.428± 0.045

6. Conclusion

This paper introduces the Trees Social Relationship Algorithm (TSR), a new metaheuristic for color image multilevel thresholding segmentation. Inspired by the social behavior of trees, TSR uses Kapur's entropy as the objective function to segment images into foreground and background regions. TSR is compared with ten other algorithms on six benchmark color images, demonstrating superior performance in fitness value, visual quality, and segmentation consistency. TSR also exhibits fast convergence and robustness to noise and contrast.

TSR offers several advantages: it handles various image types and complexities, optimizes both Kapur's and Otsu's entropy, adjusts for different values of k, and maintains image features. However, TSR's convergence rate slows with higher k values and may fall into local optima due to its simple operators. Future work should focus on adaptive population sizes and iteration numbers, enhanced crossover and mutation operators, and application to diverse image types and tasks. These improvements could address TSR's limitations and extend its applicability.

In conclusion, TSR is an effective and novel algorithm for color image segmentation, outperforming existing methods. Kapur's entropy proves superior to Otsu's for this task, and further research is encouraged to explore TSR's broader potential.

Authors Contributions

Research design and conceptualization, writing-reviewing and editing by S. F; Data gathering, computing, and writing the draft by M. R. Y.; Validation and writing-creating the initial design by M. A.
All authors have read and agreed to the published version of the manuscript.

Data Availability

Access to the data supporting this study will be available upon request to fakherisoheil@iau.ac.ir or mahmoud.alimoradi@aihe.ac.ir.

Conflict of Interest

The authors declare no conflicts of interest regarding the publication of this research.

Funding

This research received no funding.

References

1. J. Anitha, S. I. A. Pandian, and S. A. Agnes, "An efficient multilevel color image thresholding based on modified whale optimization algorithm," Expert Syst. Appl., vol. 178, p. 115003, 2021.
2. V. K. Bohat and K. V. Arya, "A new heuristic for multilevel thresholding of images," Expert Syst. Appl., vol. 117, pp. 176–203, 2019.
3. S. Pare, A. Kumar, G. K. Singh, and V. Bajaj, "Image Segmentation Using Multilevel Thresholding: A Research Review," Iran. J. Sci. Technol. Trans. Electr. Eng., vol. 44, no. 1, pp. 1–29, Mar. 2020, doi: 10.1007/s40998-019-00251-1.
4. "Multilevel image thresholding using entropy of histogram and recently developed population-based metaheuristic algorithms | SpringerLink." Accessed: Aug. 09, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s12065-017-0152-y
5. N.-D. Hoang, "Image processing-based pitting corrosion detection using metaheuristic optimized multilevel image thresholding and machine-learning approaches," Math. Probl. Eng., vol. 2020, pp. 1–19, 2020.
6. "Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in Mammography | State of the Art in Digital Mammographic Image Analysis." Accessed: Aug. 01, 2021. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789812797834_0011
7. B. Wu, J. Zhou, X. Ji, Y. Yin, and X. Shen, "An ameliorated teaching–learning-based optimization algorithm based study of image segmentation for multilevel thresholding using Kapur's entropy and Otsu's between class variance," Inf. Sci., vol. 533, pp. 72–107, 2020.
8. M. H. Merzban and M. Elbayoumi, "Efficient solution of Otsu multilevel image thresholding: A comparative study," Expert Syst. Appl., vol. 116, pp. 299–309, 2019.

9. T. Sato, Y. Ikeya, S. Adachi, K. Yagasaki, K. Nihei, and N. Itoh, “Extraction of strawberry leaves with supercritical carbon dioxide and entrainers: Antioxidant capacity, total phenolic content, and inhibitory effect on uric acid production of the extract,” Food Bioprod. Process., vol. 117, pp. 160–169, 2019.
10. M. Alimoradi, H. Azgomi, and A. Asghari, “Trees Social Relations Optimization Algorithm: A new Swarm-Based metaheuristic technique to solve continuous and discrete optimization problems,” Math. Comput. Simul., vol. 194, pp. 629–664, Apr. 2022, doi: 10.1016/j.matcom.2021.12.010.
11. “Empirical study of particle swarm optimization | IEEE Conference Publication | IEEE Xplore.” Accessed: Dec. 26, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/785511/
12. D. Karaboga and C. Ozturk, “A novel clustering approach: Artificial Bee Colony (ABC) algorithm,” Appl. Soft Comput., vol. 11, no. 1, pp. 652–657, 2011.
13. X.-S. Yang and A. Hossein Gandomi, “Bat algorithm: a novel approach for global engineering optimization,” Eng. Comput., vol. 29, no. 5, pp. 464–483, 2012.
14. C. Guo, H. Tang, B. Niu, and C. B. P. Lee, “A survey of bacterial foraging optimization,” Neurocomputing, vol. 452, pp. 728–746, 2021.
15. “Backtracking search optimization algorithm - Google Scholar.” Accessed: Dec. 26, 2023. [Online]. Available: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Backtracking+search+optimization+algorithm+&btnG=
16. S. M. Sait, A. Bala, and A. H. El-Maleh, “Cuckoo search based resource optimization of datacenters,” Appl. Intell., vol. 44, no. 3, pp. 489–506, Apr. 2016, doi: 10.1007/s10489-015-0710-x.
17. M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, “Differential Evolution: A review of more than two decades of research,” Eng. Appl. Artif. Intell., vol. 90, p. 103479, 2020.
18. V. Filipović, A. Kartelj, and D. Matić, “An electromagnetism metaheuristic for solving the Maximum Betweenness Problem,” Appl. Soft Comput., vol. 13, no. 2, pp. 1303–1313, Feb. 2013, doi: 10.1016/j.asoc.2012.10.015.
19. X.-S. Yang, “Firefly Algorithm, Stochastic Test Functions and Design Optimisation,” ArXiv10031409 Math, Mar. 2010, Accessed: Jan. 09, 2022. [Online]. Available: http://arxiv.org/abs/1003.1409
20. “Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics | IEEE Conference Publication | IEEE Xplore.” Accessed: Dec. 26, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5562213
21. S. Pare, A. Kumar, G. K. Singh, and V. Bajaj, “Image Segmentation Using Multilevel Thresholding: A Research Review,” Iran. J. Sci. Technol. Trans. Electr. Eng., vol. 44, no. 1, pp. 1–29, Mar. 2020, doi: 10.1007/s40998-019-00251-1.
22. A. Wunnava, M. K. Naik, R. Panda, B. Jena, and A. Abraham, “An adaptive Harris hawks optimization technique for two dimensional grey gradient based multilevel image thresholding,” Appl. Soft Comput., vol. 95, p. 106526, 2020.
23. A. K. Bhandari, A. Kumar, and G. K. Singh, “Tsallis entropy based multilevel thresholding for colored satellite image segmentation using evolutionary algorithms,” Expert Syst. Appl., vol. 42, no. 22, pp. 8707–8730, 2015.
24. T. Capurso, L. Bergamini, and M. Torresi, “A new generation of centrifugal pumps for high conversion efficiency,” Energy Convers. Manag., vol. 256, p. 115341, Mar. 2022, doi: 10.1016/j.enconman.2022.115341.
25. Y. Li, B. Tang, and Y. Yi, “A novel complexity-based mode feature representation for feature extraction of ship-radiated noise using VMD and slope entropy,” Appl. Acoust., vol. 196, p. 108899, Jul. 2022, doi: 10.1016/j.apacoust.2022.108899.

26. S. Singh, N. Mittal, and H. Singh, “A multilevel thresholding algorithm using LebTLBO for image segmentation,” Neural Comput. Appl., vol. 32, no. 21, pp. 16681–16706, Nov. 2020, doi: 10.1007/s00521-020-04989-2.
27. Z. K. Eisham, Md. M. Haque, Md. S. Rahman, M. M. Nishat, F. Faisal, and M. R. Islam, “Chimp optimization algorithm in multilevel image thresholding and image clustering,” Evol. Syst., vol. 14, no. 4, pp. 605–648, Aug. 2023, doi: 10.1007/s12530-022-09443-3.
28. H. Alrezaamiri, A. Ebrahimnejad, and H. Motameni, “Software requirement optimization using a fuzzy artificial chemical reaction optimization algorithm,” Soft Comput., vol. 23, no. 20, pp. 9979–9994, Oct. 2019, doi: 10.1007/s00500-018-3553-7.
29. L. Li, L. Sun, Y. Xue, S. Li, X. Huang, and R. F. Mansour, “Fuzzy multilevel image thresholding based on improved coyote optimization algorithm,” IEEE Access, vol. 9, pp. 33595–33607, 2021.
30. A. Daliri, M. Alimoradi, M. Zabihimayvan, and R. Sadeghi, “World Hyper-Heuristic: A novel reinforcement learning approach for dynamic exploration and exploitation,” Expert Syst. Appl., vol. 244, p. 122931, 2024.
31. J. Wu et al., “Medical sam adapter: Adapting segment anything model for medical image segmentation,” ArXiv Prepr. ArXiv230412620, 2023, Accessed: Jan. 05, 2024. [Online]. Available: https://arxiv.org/abs/2304.12620
32. D. Devarriya, C. Gulati, V. Mansharamani, A. Sakalle, and A. Bhardwaj, “Unbalanced breast cancer data classification using novel fitness functions in genetic programming,” Expert Syst. Appl., vol. 140, p. 112866, Feb. 2020, doi: 10.1016/j.eswa.2019.112866.
33. “Stagger PRI Radar Signal Deinterleaving based on Image Semantic Segmentation | IEEE Conference Publication | IEEE Xplore.” Accessed: Jan. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9909452
34. Z. Wang, E. Wang, and Y. Zhu, “Image segmentation evaluation: a survey of methods,” Artif. Intell. Rev., vol. 53, no. 8, pp. 5637–5674, Dec. 2020, doi: 10.1007/s10462-020-09830-9.
35. A. Vierra, A. Razzaq, and A. Andreadis, “Continuous variable analyses: T-test, Mann–Whitney U, Wilcoxon sign rank,” in Translational Surgery, Elsevier, 2023, pp. 165–170. Accessed: Jan. 05, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323903004000458
36. S. Thapliyal and N. Kumar, “A new multilevel image thresholding algorithm based on image partitioning approach with metaheuristic parameter for segmentation,” 2023, Accessed: Jan. 05, 2024. [Online]. Available: https://www.researchsquare.com/article/rs-3197892/latest
37. A. Mittal, A. K. Moorthy, and A. C. Bovik, “Blind/Referenceless Image Spatial Quality Evaluator,” in 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Nov. 2011, pp. 723–727. doi: 10.1109/ACSSC.2011.6190099.
38. J. Shim and Y. Lee, “Comparison of CT Image Performance with or without Tin Filter based on Blind Image Quality Evaluation Method,” J. Korean Soc. Radiol., vol. 15, no. 3, pp. 301–306, 2021.
39. A. Asghari, H. Azgomi, A. A. Zoraghchian, and A. Barzegarinezhad, “Energy-aware server placement in mobile edge computing using trees social relations optimization algorithm,” J. Supercomput., Oct. 2023, doi: 10.1007/s11227-023-05692-4.
40. A. Asghari, H. Azgomi, and Z. Darvishmofarahi, “Multi-objective edge server placement using the whale optimization algorithm and game theory,” Soft Comput., vol. 27, no. 21, pp. 16143–16157, Nov. 2023, doi: 10.1007/s00500-023-07995-3.
41. M. Alimoradi, M. Zabihimayvan, A. Daliri, R. Sledzik, and R. Sadeghi, “Deep Neural Classification of Darknet Traffic,” in Frontiers in Artificial Intelligence and Applications, A. Cortés, F. Grimaldo, and T. Flaminio, Eds., IOS Press, 2022. doi: 10.3233/FAIA220323.

42. A. Daliri, A. Asghari, H. Azgomi, and M. Alimoradi, “The water optimization algorithm: a novel metaheuristic for solving optimization problems,” Appl. Intell., vol. 52, no. 15, pp. 17990–18029, Dec. 2022, doi: 10.1007/s10489-022-03397-4.