An Improved Tunicate Swarm Algorithm for Global Optimization and Image Segmentation

ESSAM H. HOUSSEIN1, BAHAA EL-DIN HELMY2, AHMED A. ELNGAR2, DIAA SALAMA ABDELMINAAM3, AND HASSAN SHABAN1

1Faculty of Computers and Information, Minia University, Minia 61519, Egypt
2Faculty of Computers and Artificial Intelligence, Beni-Suef University, Beni Suef 62619, Egypt
3Faculty of Computers and Artificial Intelligence, Benha University, Benha 12311, Egypt
Corresponding author: Essam H. Houssein (essam.halim@mu.edu.eg)

ABSTRACT This study integrates a tunicate swarm algorithm (TSA) with a local escaping operator (LEO) for overcoming the weaknesses of the original TSA. The LEO strategy in TSA–LEO prevents searching deflation in TSA and improves the convergence rate and local search efficiency of swarm agents. The efficiency of the proposed TSA–LEO was verified on the CEC’2017 test suite, and its performance was compared with seven metaheuristic algorithms (MAs). The comparisons revealed that LEO significantly helps TSA by improving the quality of its solutions and accelerating the convergence rate. TSA–LEO was further tested on a real-world problem, namely, segmentation based on the objective functions of Otsu and Kapur. A set of well-known evaluation metrics was used to validate the performance and segmentation quality of the proposed TSA–LEO. The proposed TSA-LEO outperforms other MA algorithms in terms of fitness, peak signal-to-noise ratio, structural similarity, feature similarity, and segmentation findings.

INDEX TERMS Metaheuristic algorithms, tunicate swarm algorithm (TSA), local escaping operator (LEO), multilevel thresholding, image segmentation, Kapur’s entropy, Otsu method.

I. INTRODUCTION
Objective optimization problems, such as minimizing time consumption, energy, cost, and error or maximizing efficiency, performance, and quality of a process, are commonly encountered in real-world applications [1]. Recently, several researchers have embraced a new family of optimization algorithms called metaheuristic algorithms (MAs), and numerous optimizers have been developed for complex real-world problems. Such algorithms randomly search the feature space to obtain an optimal solution among various solutions, which are mainly inspired by nature. Among a large body of nature-inspired MAs, some are popular such as moth flame optimization (MFO) [2], whale optimization algorithm (WOA) [3], sine cosine optimization (SCA) [4], seagull optimization algorithm (SOA) [5], krill herd algorithm [6], and barnacles mating optimizer (BMO) [7], because they are simple, efficient, and robust in finding optimal solutions. Moreover, the No-Free Lunch Theorem [8] states that no specific optimization algorithm can accurately solve multiple optimization problems. Thus, several MAs have been developed for use in biomedicine [9], [10], bioinformatics [11], [12], cheminformatics [13], [14], feature selection [15], engineering problems [16]–[19], pattern recognition, text clustering [20], [21], and wireless sensor networks [22], [23]. However, all MAs need to balance exploration and exploitation stages; otherwise, solutions tend to become trapped in local optima or cannot properly converge [24], [25]. Randomization during the solution-finding process can cause such problems. Hybridization of multiple concepts from different scientific fields is mandatory, especially in human-aided systems. Hybridization can combine the advantages of different algorithms to produce enhanced versions with promising performance and accuracy.

For example, the authors in [26] improved the grey wolf optimization (GWO) algorithm for engineering design problems. The enhanced version, which is known as I-GWO, adopts a new movement strategy called dimension-learning hunting (DLH). DLH enhances the diversity of solutions to balance exploration and exploitation phases and avert local optima. Results confirmed the robustness of I-GWO on the CEC’2017 test suite functions. Moreover, the study in [27] boosted the WOA algorithm (one of the most well-known optimization algorithms) with two search strategies: chaotic...
and Gaussian mutation. The two search strategies were expected to avoid local optima by balancing the exploration and exploitation phases. The algorithm achieved promising performance results compared with state-of-the-art methods. Moreover, the algorithm proposed a method that uses static single assignment (SSA) and particle swarm organization (PSO) to solve complex optimization problems. This method prevents local optima trapping and unbalanced exploitation in the original SSA. The proposed SSA–PSO outperformed competing methods in a comparison test on the CEC’2005 and CEC’2017 functions. The authors of [28] integrated SCA with PSO, which overcomes the drawbacks of SCA in the exploitation phase. The combined ASCA–PSO achieved good performance (high accuracy and low time complexity) on several benchmarks. The authors of [29] combined SCA with operator-based linearization (OBL), increasing the performance and improving solutions of SCA. The superiority of the proposed SCA–OBL was evaluated in several benchmark functions and engineering problems. Orthodox learning strategy was hybridized with MFO to optimize its parameters [30]. This new MFO version avoids the searchability of the original MFO and enhances the diversity of solutions. The algorithm explores new regions in the search for an optimal agent with the best solution. The effectiveness of the enhanced MFO was verified on CEC’2014 test functions and several engineering problems. The proposed method outperformed other optimization algorithms as proven by the comparison result.

On the other hand, the development and application of vision systems have accelerated in the recent era [31]–[34]. Image processing without a vision system is ill-advised, and a proper pre-processing improves the accuracy of the results. Segmentation pre-processing facilitates the representation and analysis of images [35], and must be accurately performed in any vision application [36]. In particular, the image should be subdivided to extract only the regions carrying useful information. Segmentation methods can be parametric or non-parametric [37]. Parametric segmentation defines each class based on the probability density function; non-parametric segmentation uses specific standards, such as variance, entropy, or error rate, to obtain the optimal thresholds that effectively separate the image. One of the most popular and promising segmentation tools, i.e., thresholding, divides the image into multiple homogeneous segments. Thresholding is also adopted in image analysis and processing because it is easily understood and implemented [38].

Bi-level (BT) and multilevel (MT) thresholding techniques can be used to select the thresholds in a grayscale image [39]. The BT technique divides the entire image into two classes based on a single threshold state, whereas the MT technique segments the image into several classes based on two or more thresholds [40], [41]. Otsu’s between-class variance [42], Kapur’s entropy [43], and Tsallis entropy [44] are used to optimize the threshold(s). These methods have optimal thresholds that separate an image into multiple segments, but this can be considered a complex task, especially when the number of thresholds increases [45]–[47]. Several optimization methods have been blended with classical thresholding methods to operate with the complexities of multilevel thresholding problems. Tunicate swarm algorithm (TSA) is a new robust search method inspired by the strange behavior of tunicates (a marine invertebrate animal) during foraging [48].

Tunicates adopt two main strategies while searching for food: jet propulsion and swarm intelligence. Most optimization algorithms obtain new solutions based on the previous solution. Two strategies in the original TSA are used to dictate a new solution: jet propulsion and swarm intelligence. These strategies are randomly applied to the current solutions to obtain the best solution. In some optimization cases, the original TSA determines the optimal solution from subregions, which lowers the convergence rate and prevents full coverage of the search space (the latter problem leads to premature convergence of the TSA). These problems are common in most optimization algorithms, especially in complex and high-dimensional problems [49]. Local escaping operator (LEO) is, a new mathematical approach [50] that was developed as a local search used for generating an efficient solution aiming to visit the unseen search regions, and thus, escaping from the local optimal problem. Moreover, operators such as $p1$, $f1$ and $f2$ are used to balance between the exploration and exploitation phases, shown in Eqs. (9 and 8).

**MOTIVATION AND CONTRIBUTIONS:**
To mitigate TSA’s problems, this paper hybridizes the original TSA with an efficient operator LEO to address the shortcoming that the standard TSA may exhibit, i.e. 1) evades trapping in local optima, 2) balances between exploration and exploitation, and 3) improves the convergence speed. The proposed method was validated on the CEC’2017 benchmark functions, and its performance was compared with those of seven established optimization algorithms namely MFO, WOA, SCA, SOA, BMO, chaotic TSA (CTSA), and the original TSA. Then it is applied to tackle multilevel thresholding image segmentation problems based on maximizing two objective functions namely Otsu and Kapur objective functions. Peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and feature similarity (FSIM) are three quality metrics used to evaluate segmentation results in terms of fitness. Optimization and segmentation results revealed the robustness of the proposed TSA–LEO compared with a set of well-known optimization algorithms. In summary, the major contributions of the paper are summarized below:

- An efficient TSA based on LEO called TSA-LEO is presented.
- TSA-LEO is proposed for solving optimization and multilevel thresholding image segmentation.
- The effectiveness of TSA-LEO is assessed on the CEC’2020 suite.
- Two objective functions, Kapur and Otsu, are applied.
• The quality of segmentation is verified in terms of the PSNR, SSIM, FSIM.
• The proposed method is compared with state-of-the-art algorithms.
• Extensive results show the more stable performance of the proposed TSA-LEO.
• Significant threshold results are obtained.

The remainder of this paper is arranged as follows. Section II devices the problem; Section III introduces the proposed TSA–LEO and its main procedure; Section IV discusses and analyzes the benchmark results; and in Section V, TSA–LEO is applied to image segmentation-based thresholding. Conclusions and forthcoming works are represented in Section VI.

II. PRELIMINARIES

A. TUNICATE SWARM ALGORITHM (TSA)

Kaur et al. [48] proposed a bio-inspired optimization algorithm that simulates the natural foraging process of tunicates, marine invertebrates that emit bright bio-luminescence. The TSA was inspired by the strange behaviors of tunicates in oceans, in particular, the jet-drive and swarm intelligence of their foraging process. A mathematical model of jet propulsion is developed under three constraints: preventing conflict among the exploration agents, following the positions of the most qualified agents, and remaining near the optimal agents.

1) PREVENTING CONFLICTS AMONG THE AGENTS

To prevent inter-agent conflicts while searching for better positions, the new agent positions are calculated as:

\[
\vec{A} = \frac{\vec{G}}{M} \\
\vec{G} = c_2 + c_3 - \vec{F} \\
\vec{F} = c_1 \cdot \vec{F}
\]

where \( \vec{A} \) is a vector of new agent positions, \( \vec{G} \) is the gravity force, \( \vec{F} \) represents the water flow in the deep ocean, and \( c_1, c_2 \) and \( c_3 \) are three random numbers. The social forces between agents are stored in a new vector \( \vec{M} \), represented as follows:

\[
\vec{M} = [P_{min} + c_1 \cdot P_{max} - P_{min}].
\]

Here \( P_{min} = 1 \) and \( P_{max} = 4 \) describe the first and second subordinates respectively, indicating the speeds of establishing social interactions.

2) FOLLOWING THE POSITIONS OF THE BEST AGENT

Following the current best agent is essential for reaching the optimal solution. Hence, after ensuring that no conflicts exist between neighboring agents in the swarm, the best position of the best agent is computed as,

\[
\vec{PD} = |X_{best} - r_{rand} \cdot P_p(x) |
\]

where \( \vec{PD} \) stores the length between the food origin and the optimal agent, \( X_{best} \) is the best position, \( r_{rand} \) is a stochastic value in the range \([0, 1]\), and the vector \( P_p(x) \) contains the positions of the tunicates during iteration \( x \).

3) KEEPING CLOSE TO THE OPTIMAL AGENTS

To ensure that search agents still close to the best agent, their positions are computed as follows:

\[
P_p^*(x) = \begin{cases} 
X_{best} + A \cdot \vec{PD} & \text{if } r_{rand} \geq 0.5 \\
X_{best} - A \cdot \vec{PD} & \text{if } r_{rand} < 0.5
\end{cases}
\]

where \( P_p^*(x) \) contains the updated positions of the agents at iteration \( x \) relative to the best scored position \( X_{best} \).

4) SWARMING BEHAVIOR

To model the swarming behavior of tunicates, the positions of the current agents are updated based on the positions of two agents:

\[
P_p(x + 1) = \frac{P_p^*(x) + P_p(x + 1)}{2 + c_1}
\]

To clarify the TSA, the main steps given below illustrate the flow of the original TSA in detail.

Step 1: Initialize the first population of tunicates \( P_p \).

Step 2: Set the original value for parameters and the highest number of iterations.

Step 3: Measure the fitness value of each exploration agent.

Step 4: After calculating the fitness, the best agent is investigated in the supplied search space.

Step 5: Update the positions of each exploration agent using Eq7.

Step 6: Return the new updated agents to its boundaries.

Step 7: Measure the fitness cost of the updated search agent. If there is a better solution than the past optimal solution, then update \( P_p^* \) and save the best solution in \( X_{best} \).

Step 8: If the termination criterion is met, then the processes stop. Otherwise, iterate Steps 5–8.

Step 9: Declare the best optimal solution (\( X_{best} \)), which is achieved so far.

B. LOCAL ESCAPING OPERATOR (LEO)

The LEO proposed as a local search algorithm in [50] which is used to enhance the ability of an optimization algorithm namely Gradient-based optimizer (GBO) aiming to explore new regions which are desired in complex real-world problems. The LEO enhances the quality of solutions by updating their positions under some criteria. Specifically, it prevents the algorithm from trapping in local optima and improves its convergence behavior. LEO generates its alternative solutions \( P_{LEO} \) with excellent performance by using several solutions such as the best position of tunicates \( X_{best} \), two randomly generated solutions \( X_1 \) and \( X_2 \), two randomly chosen solutions \( X_1^m \) and \( X_2^m \), and a new randomly generated solution \( X_k^m \). Hence, the solution \( P_{LEO} \) can be determined based on
Eqs. (8 and 9) which can be mathematically formulated as follows:

\[
\begin{align*}
\text{if rand} & < pr \\
\text{if rand} & < 0.5
\end{align*}
\]

\[
\tilde{P}_n^{m+1} = \tilde{P}_n^{m} + f_1 \times (u_1 \times X_{\text{best}} - u_2 \times X_n^m) + f_2 \times \rho_1 \times (u_3 \times (X_{n}^m - X_1^m) + u_2 \times (X_{r_1}^m - X_{r_2}^m)) / 2
\]

(8)

\[
P_{p}^{m+1} = P_{LEO} \text{ Else }
\]

\[
\tilde{P}_n^{m+1} = P_{LEO}
\]

Here \( \tilde{P}_n \) is the current tunicate position, \( X_{\text{best}} \) is the best scored position, \( pr \) is the probability of performing LEO strategy where \( pr = 0.3 \), \( rand \) represents a random value in range \( \in [0, 1] \). \( f_1 \) and \( f_2 \) are uniformly distributed random values \( \in [-1, 1] \). \( X_1^m \) and \( X_n^m \) represent two random solutions chosen from the population, \( X_1 \) and \( X_2 \) are two solutions which are randomly generated as shown in Eq10 from the current population.

\[
X_1^m \times X_n^m = LB + rand(\text{Dim}) \times (UB - LB)
\]

(10)

where LB, UB are the lower and upper bounds, Dim is the dimension of any solution. Moreover, \( n \) and \( m \) represent the coordinates of the solution \( (n = 1, 2, 3, \ldots, N) \) and \( (m = 1, 2, 3, \ldots, \text{Dim}) \). In addition, \( u_1, u_2, \) and \( u_3 \) are three variables that are randomly generated as following:

\[
\begin{align*}
u_1 & = L_1 \times 2 \times rand + (1 - L_1) \\
u_2 & = L_1 \times rand + (1 - L_1) \\
u_3 & = L_1 \times rand + (1 - L_1)
\end{align*}
\]

(11)-(13)

where \( L_1 \) is a binary parameter \( (L_1 = 1 \text{ if } \mu_1 < 0.5, 0 \text{ otherwise}) \). \( \mu_1 \) is a number in the range of \( [0, 1] \).

Moreover \( \rho_1 \) is introduced to balance the exploration and exploitation searching processes, and it can be expressed as:

\[
\rho_1 = 2 \times \text{rand} \times \alpha - \alpha
\]

(14)

\[
\alpha = \left| \beta \times \sin \left( \frac{3\pi}{2} + \sin \left( \beta \times \frac{3\pi}{2} \right) \right) \right|
\]

(15)

\[
\beta = \beta_{\text{min}} + (\beta_{\text{max}} - \beta_{\text{min}}) \times \left( 1 - \left( \frac{t}{\text{Maxiterations}} \right) \right)^2
\]

(16)

where \( \beta_{\text{min}} \) and \( \beta_{\text{max}} \) are set to 0.2 and 1.2 respectively, \( t \) is the current iteration, and \( \text{Maxiterations} \) is the maximum number of iterations. To balance the exploration and exploitation processes, parameter \( \rho_1 \) changes based on the sine function \( \alpha \).

To determine the solution \( X_k^m \) in Eq. (28), the following scheme is suggested:

\[
X_k^m = \begin{cases} 
    x_{rand} & \text{if } \mu_2 < 0.5 \\
    x_p^m & \text{otherwise}
\end{cases}
\]

(17)

where \( x_{rand} \) is a new solution that can be calculated as shown in Eq18, \( x_p^m \) is a random solution selected from the population \( (p \in \{1, 2, \ldots, N\}) \), \( \mu_2 \) is a random number in the range of [0, 1].

\[
x_{rand} = \text{X}_{\text{min}} + \text{rand}(0, 1) \times (\text{X}_{\text{max}} - \text{X}_{\text{min}})
\]

(18)

Eq17 can be simplified as follows:

\[
X_k^m = L_2 \times x_p^m + (1 - L_2) \times x_{rand}
\]

(19)

where \( L_2 \) is a binary parameter with a value of 0 or 1. If parameter \( \mu_1 \) is less than 0.5, the value of \( L_1 \) is 1, otherwise, it is 0.

**III. THE PROPOSED TSA-LEO**

This section illustrates the implementation of the proposed TSA–LEO method to improve the ability of the original TSA by allowing it to visit promising regions. LEO is specifically used to improve the performance of the best solutions of the original TSA. The TSA–LEO algorithm follows the main steps of the original TSA, and employs the LEO operator to encourage the visitation of new regions. LEO improves the search for global optima and convergence rate of the algorithm, dynamically evading stagnation in local optima. In the following section, the implementation of the proposed TSA–LEO is given in detail.

**A. PRIMITIVE STEP OF TSA-LEO**

The proposed TSA–LEO method, like numerous other optimization algorithms, begins by randomly initializing its parameters, \( \hat{A}, \hat{G}, \hat{F}, \hat{M} \) as shown in Eqs. 1 to 4, respectively. Moreover, creating the initial population \( \tilde{P}_p \) as shown below.

\[
\tilde{P}_p = LB + rand(N, \text{Dim}) \times (UB - LB)
\]

(20)

where \( \tilde{P}_p \) is the initial population, and \( N \) denotes the number of random solutions \( i \in \{1, 2, \ldots, N\} \) each solution is limited between the upper and lower boundaries \( (UB \text{ and } LB) \) with a dimension of \( \text{Dim} \) in the search space.

**B. UPDATING SOLUTION SCENARIOS**

The position updating process is conducted based on two scenarios. First, generating a two-agent solution as shown in Eq. 7, or based on the best position obtained so far using Eq 6 and saving results. In this step, the original TSA is executed conventionally. In the second scenario, to the solution is updated using the LEO strategy to improve the solution efficiency. The LEO distinction between two paths depends on a specific condition as shown in Eqs. 8, and 9. If \( rand < 0.5 \), the first path is selected to perform the process of solution updating as shown in Eq8; otherwise, the second path Eq.9 is selected to find the new solution.
**C. OPTIMIZATION SCENARIOS**

This step is performed to evaluate the vector of solutions generated from the previous phase in each iteration to enhance the quality of the further solutions. Accordingly, TSA–LEO computes the fitness value \( f(\vec{P}_p) \) of each tunicate position in the current population. The best-scoring solution \( X_{best} \) is then determined, saved, and extracted at the updating stage.

**D. TERMINATION CRITERIA**

After completing the optimization scenarios and iterating until reaching the stopping criteria, the proposed TSA–LEO retrieves the optimal solution according to the best fitness. Algorithm 1 gives the pseudo code of the TSA–LEO algorithm, and a detailed flowchart is shown in Fig. 1.

### Algorithm 1: The Proposed TSA–LEO Algorithm

```plaintext
procedure TSA–LEO
    Initialize the first population \( \vec{P}_p \) randomly.
    while \( x < Max_{iterations} \) do
        for \( i = 1 \) to \( N \) do
            \( X_{best} \leftarrow \text{Best}(f(\vec{P}_p)) \)
            /*Jet propulsion behavior*/
            Calculate the parameters \( \vec{A}, \vec{G}, \vec{F}, \vec{M} \), and \( \vec{PD} \) using Eqs.(1 to 5) respectively.
            /*Swarm behavior*/
            if \( r_{rand} \geq 0.5 \) then
                \( \vec{P}_p \leftarrow \vec{P}_p + X_{best} + \vec{A} \times \vec{PD} \)
            else
                \( \vec{P}_p \leftarrow \vec{P}_p + X_{best} - \vec{A} \times \vec{PD} \)
            end if
        end for
        /*Local escaping operator (LEO)*/
        if \( rand < pr \) then
            if \( rand < 0.5 \) then
                Update \( \vec{P}_p \) using Eq8
            else
                Update \( \vec{P}_p \) using Eq9
            end if
        end if
        \( X_{best} \leftarrow \text{Best}(f(\vec{P}_p)) \)
        \( x \leftarrow x + 1 \)
    end while
    return \( X_{best} \)
end procedure
```

**E. COMPUTATIONAL COMPLEXITY OF THE PROPOSED TSA–LEO**

This subsection reports and estimates the computational complexity of the proposed TSA–LEO algorithm in terms of time and space complexities.

1) **TIME COMPLEXITY**

TSA–LEO starts by creating an initial population of size \( N \) for each problem dimension \( \text{Dim} \), such that the complexity of initialization is \( O(N \times \text{Dim}) \) time complexity. Furthermore, TSA–LEO computes the fitness of each population, so the complexity of this process is \( O(\text{Max}_{iterations} \times N \times \text{Dim}) \), where \( \text{Max}_{iterations} \) denotes the maximum number of iterations. In addition, TSA–LEO needs \( O(T) \) time complexity to perform its main processes, where \( T \) represents the number of jet propulsion, swarm behaviors, and the LEO processes. The overall time complexity of the proposed TSA–LEO can be represented by \( O(\text{Max}_{iterations} \times T \times N \times \text{Dim}) \).

2) **SPACE COMPLEXITY**

Space complexity defines the total amount of space occupied by the algorithm. Now, TSA–LEO takes \( O(N \times \text{Dim}) \) space complexity.

**IV. PERFORMANCE EVALUATION OF TSA–LEO**

**A. PARAMETER SETTINGS**

To accurately evaluate the effectiveness of the proposed TSA–LEO, the algorithm was competed against seven other algorithms, namely, MFO [2], WOA [3], SCA [4], SOA [5], BMO [7], CTSA, and the original TSA. Each method was executed 30 times through (at most) 1000 iterations. The user population size was set to 30. The parameters of each algorithm were set to the values of the first-published standard versions. Table 1 lists the parameters and setting positions of TSA–LEO.

**B. DEFINITION OF CEC’17 TEST SUITE FUNCTIONS**

The CEC’17 test suite was selected as a test problem because it has high complexity and is customized for global optimization. The CEC’17 test suite contains 30 functions, but function F2 was excluded because to its instability. Therefore, the used benchmark contained 29 test functions. The test suite contains 29 functions and is composed of unimodal shifted and rotated functions; multimodal shifted and rotated functions; hybrid functions; and composition functions as shown in [51].

Fig. 2 shows the landscapes of 16 selected functions in two-dimensional space and provides an intuitive understanding of the functional differences and nature of the problems.

**C. STATISTICAL RESULTS ANALYSIS**

The CEC’17 benchmark functions are employed to assess the performance of advanced TSA–LEO. Mean and standard deviation (STD) values of each run’s best solutions are used to measure the algorithm efficiency. Table 2 represents the mean and STD obtained from the proposed TSA–LEO and other comparative algorithms for each CEC’17 function with 50-dimension; the best results (minimum values) are highlighted in bold. Regarding the best optimal fitness results shown in Table 2, the proposed TSA–LEO algorithm gains the best fitness results in 20 functions (0 unimodal functions, 5 multimodal functions, 8 in hybrid functions, and 7 in composition functions) gaining the first rank with overall ratio (69%) of test functions, whereas the CTSA algorithm gains...
the best fitness of 9 functions gaining the second rank with overall ratio (31%) of test functions, while the other competing algorithms fail to gain the best fitness in any test function. This means that the proposed TSA-LEO algorithm can effectively solve multimodal functions (F4 to F10), hybrid functions (F11 to F20), and composition functions (F21 to F30). Table 3 shows the rank-sum results for fitness according to Wilcoxon rank-sum test. After applying the rank-sum test between the proposed TSA-LEO algorithm and each of the other algorithms (MFO, WOA, SCA, SOA, BMO, TSA, CTSA, and TSA-LEO) a difference between all competitors in contrast to the proposed TSA-LEO is noticed. TSA-LEO vs MFO has a significant difference with a ratio of (96.55%), TSA-LEO vs WOA has a significant difference with a ratio of (100%), TSA-LEO vs SCA has a significant difference with a ratio of (93.10%), TSA-LEO vs SOA has a significant difference with a ratio of (86.20%), TSA-LEO vs BMO has a significant difference with a ratio of (100%), TSA-LEO vs TSA has a significant difference with a ratio of (82.75%), TSA-LEO vs CTSA has a significant difference with a ratio of (82.75%); this means that the proposed TSA-LEO algorithm has a significant development. Moreover, based on

**TABLE 1.** Parameter settings of TSA-LEO and competing algorithms.

| Algorithms   | Parameters setting                                      |
|--------------|---------------------------------------------------------|
| Common Settings | Population size: $N = 30$                               |
|              | Maximum iterations: $t_{max} = 1000$                   |
|              | Problem dimensions $Dim = 50$                          |
|              | Number of independent runs 30                          |
| MFO          | $\beta = 1$ and $\alpha$ decreases linearly from $-1$ to $-2$ (Default) |
| WOA          | $\alpha$ variable decreases linearly from 2 to 0 (Default) |
|              | $\alpha 2$ linearly decreases from -1 to -2 (Default)   |
| SCA          | $A = 2$ (Default)                                      |
| SOA          | $A = [0, 2]$ and $f_c = 2$ (Default)                   |
| BMO          | $pl = 7$ (Default)                                     |
| TSA and CTSA | $P_{min} = 1$ and $P_{max} = 4$                        |
| TSA-LEO      | $P_{min} = 1$, $P_{max} = 4$ and $pr = 0.3$            |
Friedman’s mean rank test results, the proposed TSA–LEO ranks first compared to the other algorithms. Overall statistical results showed that in solving different advanced benchmarks, the proposed method was more effective than other well-known optimization methods.

**D. BOXPLOT BEHAVIOR ANALYSIS**

Data distribution characteristics can be displayed by boxplot analysis. Boxplots are efficient for depicting data distributions into quartiles. The minimum and maximum edges of the whiskers are the lowest and largest data points reached by the algorithm. The ends of the rectangles define the lower and upper quartile. A narrow boxplot signifies a high agreement between data. Due to space limitations, Fig. 6 illustrates 15 functions. Figure 3 shows the analyses of F1–F30 functions boxplot for Dim = 50. For most functions, boxplots of the proposed TSA–LEO algorithm are narrow compared with other algorithm distributions and thus have the lowest values. Indeed, the proposed TSA–LEO algorithm performs better than other algorithms for most of the test functions.

**E. CONVERGENCE BEHAVIOR ANALYSIS**

Fig. 4 shows the convergence plots of MFO, WOA, SCA, SOA, BMO, TSA, and TSA–LEO on the CEC’17 functions. The proposed TSA–LEO achieved (near)-optimal solutions and fast convergence on most functions; hence, it can solve problems requiring fast computation, such as online problems. Furthermore, the algorithm exhibited stable behavior, and its solutions smoothly converged in most of the tested problems. Due to space limitations, Fig. 4 illustrates 15 functions.

**F. EXPLORATION AND EXPLOITATION ANALYSIS**

Fig. 5 records exploration-exploitation ratios during search maintained by the proposed TSA-LEO while solving a set of CEC’17 test functions with dimension of 30. From Fig. 5 it is noticeable that the proposed TSA-LEO starts with a high exploration ratio and low exploitation ratio, but mostly later transformed into exploitation strategy during most of the iterations in most of the CEC’17 functions. This indicates that the proposed TSA-LEO balances effectively between exploration and exploitation phases.

**G. QUALITATIVE METRICS ANALYSIS**

Fig. 6 shows the collective behavior of foraging tunicates. The first pillar represents a set of the CEC’17 functions as shapes in two-dimensional space. The second pillar illustrates the search history of the tunicates, show-
TABLE 2. Mean and STD of the fitness values of different optimization algorithms, obtained from 30 runs of 50-dimensional CEC’17 functions.

| Function | MFO | WOA | SCA | SCA | BMO | TSA | TSA–LEO |
|----------|-----|-----|-----|-----|-----|-----|--------|
| F1       | 1.286±1 | 2.358±0.8 | 2.385±0.9 | 2.385±0.9 | 4.038±0.1 | 4.038±0.1 | 8.154±0.9 |
| F3       | 0.160±1 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 |
| F5       | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 |
| F7       | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 |
| F9       | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F11      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F13      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F15      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F17      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F19      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |

TABLE 3. Comparison of the p-values obtained from the Wilcoxon signed-rank test between the pairs of TSA–LEO vs. MFO, TSA–LEO vs. WOA, TSA–LEO vs. SCA, TSA–LEO vs. SOA, TSA–LEO vs. BMO, TSA–LEO vs. TSA, and TSA–LEO vs. CTSA for CEC’17 test functions in terms of Fitness results.

| Function | MFO | WOA | SCA | SCA | BMO | TSA | TSA–LEO |
|----------|-----|-----|-----|-----|-----|-----|--------|
| F1       | 5.064±1 | 5.064±1 | 5.064±1 | 5.064±1 | 5.064±1 | 5.064±1 | 5.064±1 |
| F3       | 0.160±1 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 | 0.320±0.5 |
| F5       | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 | 2.123±1 |
| F7       | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 | 0.047±1 |
| F9       | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F11      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F13      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F15      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F17      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |
| F19      | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 | 0.534±1 |

VOLUME 9, 2021

56073

E. H. Houssein et al.: Improved TSA for Global Optimization and Image Segmentation

ing their exploitation behavior to achieve the desired results. The third pillar displays the average fitness over 100 iterations, explaining how diversified new agents assist in the search of the best solution. The proposed TSA–LEO can find the areas with the best fitness for most functions according to the search history pillar. In terms of average fitness history, all curves are decreasing, which means that the population improves at each iteration.
This constant improvement substantiates a collaborative searching behavior and supports the efficiency of updating particle law. Finally, convergence curve and optimization history revealed the progress of fitness over several iterations. The decrease in optimization history indicates that the solutions are optimized during iterations until reaching the optimal solution.

V. EXPERIMENTAL RESULTS AND ANALYSIS

This section employs the proposed TSA–LEO to solve thresholding-based image segmentation problems. In this evaluation, TSA–LEO was expected to select the thresholds that best segmented a set of benchmark images by maximizing a well-known thresholding technique, namely, Otsu’s objective function.
A. MULTI-THRESHOLDING IMAGE SEGMENTATION STUDIES

In this research, image thresholding shows the efficiency of metaheuristic algorithms in the relevant method [35], [47], [52]. In this regard, there are numerous examples of meta-heuristic applications; however, a few prominent state-of-the-art research works are given. To tackle the problems of multi-thresholding, Upadhyay and Chhabra [53] used the crow search algorithm (CSA) to maximize Kapur’s method. The proposed model was compared with a set of
FIGURE 5. The Graphical representation of the exploration and exploitation phases for the proposed TSA-LEO over the CEC’17 functions with $\text{Dim} = 50$.

well-known metaheuristic algorithms, namely, PSO, DE, GWO, MFO, and CSA. The authors chose CSA because of its balance between exploration and exploitation, as well as less parameters to tune. Through most commonly used evaluation metrics, the authors contended to have achieved comparatively better results when tested on a set of benchmark images using multiple threshold values. Despite the success in this work, CSA has a slow convergence. Khairuzzaman and Chaudhury [54]used GWO to produce efficient image-segmentation results while finding the optimal set of
FIGURE 6. Qualitative metrics on F1, F4, F6, F12, F21, and F28: 2D views of the functions, search history, average fitness history, and optimization history.
thresholds using Otsu’s and Kapur’s functions. GWO converged to better optimum solutions than bacterial foraging optimization (BFO) and PSO; however, the proposed algorithms also posed certain disadvantages: a) its efficiency reduced when employed on noisy images and b) GWO was slower than PSO regarding the computational time. The research maintained a major weakness; it did not provide a comprehensive comparison with other well-known and established metaheuristic algorithms, but merely used PSO and BFO for comparison. To optimize threshold values for multilevel image thresholding, a modified grasshopper optimization algorithm (GOA) with Lévy flight was introduced based on Tsallis cross-entropy as the objective function [55]. The proposed model was tested on benchmark images and plant stomata. Compared with standard GOA, WOA, flower pollination algorithm (FPA), PSO, and bat algorithm (BA), the proposed GOA variant produced better segmentation accuracy with enhanced multilevel segmentation convergence on energy-based Tsallis entropy. One limitation of this study is that it did not experiment with relatively increased thresholds for high-dimensional optimization problems.

The study in [56] used the EO algorithm and Kapur’s entropy as objective function to achieve the optimal threshold values for grayscale images. To achieve enhanced search ability, the researchers improved EO with adaptive parameters. The proposed method was evaluated using several solution quality metrics such as the signal-to-noise ratio, structured similarity index, some accuracy measures like mean absolute error, and the computation time for resource complexity. The proposed EO outperformed WOA, BA, SCA, SSA, harris hawk’s optimizer (HHO), CSA, and PSO techniques.

The significance of this study can be determined with the level of thresholds used in the experiment. The researchers used up to 50 threshold levels. However, the proposed EO variant comparatively underperformed considering standard

---

**TABLE 4.** Results after applying TSA-LEO on Kapur to the set of benchmark images.

| Image    | Level = 2 | Level = 3 | Level = 4 | Level = 5 |
|----------|-----------|-----------|-----------|-----------|
| Cameraman| ![Image](image1.png) | ![Image](image2.png) | ![Image](image3.png) | ![Image](image4.png) |
| Lena     | ![Image](image5.png) | ![Image](image6.png) | ![Image](image7.png) | ![Image](image8.png) |
| Baboon   | ![Image](image9.png) | ![Image](image10.png) | ![Image](image11.png) | ![Image](image12.png) |
| Hunter   | ![Image](image13.png) | ![Image](image14.png) | ![Image](image15.png) | ![Image](image16.png) |
| Airplane | ![Image](image17.png) | ![Image](image18.png) | ![Image](image19.png) | ![Image](image20.png) |

---

**TABLE 5.** Results after applying TSA-LEO on Kapur to the set of benchmark images.

| Image    | Level = 2 | Level = 3 | Level = 4 | Level = 5 |
|----------|-----------|-----------|-----------|-----------|
| Pepper   | ![Image](image21.png) | ![Image](image22.png) | ![Image](image23.png) | ![Image](image24.png) |
| Living-Room | ![Image](image25.png) | ![Image](image26.png) | ![Image](image27.png) | ![Image](image28.png) |
| Woman    | ![Image](image29.png) | ![Image](image30.png) | ![Image](image31.png) | ![Image](image32.png) |
| Bridge   | ![Image](image33.png) | ![Image](image34.png) | ![Image](image35.png) | ![Image](image36.png) |
| Butter-Fly | ![Image](image37.png) | ![Image](image38.png) | ![Image](image39.png) | ![Image](image40.png) |
deviation values and computational time. HHO is another recent metaheuristic technique that was implemented in a similar domain using Otsu’s and Kapur’s objective functions [57]. Comparisons of the proposed method with PSO, DE, harmony search (HS), ABC, and SCA, show that it produced efficient results in terms of quality, consistency, and accuracy. However, the results of HHO were also compared with two machine learning techniques, K-means and fuzzy IterAg, revealing that these techniques performed the least in the overall image-segmentation exercise. Another limitation of this study is that it was not evaluated on color images, and the number of thresholds was manually set. Meanwhile, Díaz-Cortés et al. [58] resolved the problem of unclear regional borders in low-resolution thermography images in health-care using the dragonfly algorithm (DA). In addition, the DA technique is used to find optimum threshold values for energy curves in thermal images for breast cancer diagnosis. Based on the objective functions of Otsu’s and Kapur’s, the authors evaluated solution quality and found that DA outperformed GA, PSO, runner-root algorithm and krill-herd algorithm on a set of eight images retrieved from the DA-Breast Thermography database.

To improve the optimal threshold selection in this study, the proposed TSA-LEO algorithm was integrated with Otsu’ and Kapur’s objective functions.

\[ h_i = \frac{h_i}{NP} \sum_{i=1}^{NP} Ph_i = 1 \]  

(21)

where \( i \) is an intensity level defined in the range of \( 0 \leq i \leq L - 1 \). NP is the total number of pixels in an image. \( h_i \) denotes the number of occurrence of intensity \( i \) in the image represented by the histogram. The histogram is normalized in a probability distribution \( Ph_i \). Based on the probability distribution or threshold value \((th)\), the classes are computed for bi-level segmentation as follows:

\[ C_1 = \frac{Ph_1}{\omega_0(th)}, \ldots, \frac{Ph_{th}}{\omega_0(th)} \text{ and } C_2 = \frac{Ph_{th+1}}{\omega_1(th)}, \ldots, \frac{Ph_L}{\omega_1(th)} \]  

(22)

where \( \omega_0(th) \) and \( \omega_1(th) \) are cumulative probability distributions for \( C_1 \) and \( C_2 \), as it is shown by Eq. (23).

\[ \omega_0(th) = \sum_{i=1}^{th} Ph_i \text{ and } \omega_1(th) = \sum_{i=th+1}^{L} Ph_i \]  

(23)

It is mandatory to find the average intensity levels \( \mu_0 \) and \( \mu_1 \) that define the classes using Eq. (24). Once those values are calculated, the Otsu based between-class \( \sigma_B^2 \) is calculated using Eq. (25).

\[ \mu_0 = \sum_{i=1}^{th} iPh_i \text{ and } \mu_1 = \sum_{i=th+1}^{L} iPh_i \]  

(24)

\[ \sigma_B^2 = \sigma_1 + \sigma_2 \]  

(25)

Notice that \( \sigma_1 \) and \( \sigma_2 \) in Eq. (25) are the variances of \( C_1 \) and \( C_2 \) which are defined as follow:

\[ \sigma_1 = \omega_0(\mu_0 + \mu_T)^2 \text{ and } \sigma_2 = \omega_1(\mu_1 + \mu_T)^2 \]  

(26)

where \( \mu_T = \omega_0\mu_0 + \omega_1\mu_1 \) and \( \omega_0 + \omega_1 = 1 \) based on the values \( \sigma_1 \) and \( \sigma_2 \). Eq. (27) presents the objective function. Therefore, the optimization problem is reduced to find the intensity level that maximizes Eq. (27).

\[ F_{otsu}(th) = \max(\sigma_B^2(th)) \text{ where } 0 \leq th \leq L - 1 \]  

(27)

where \( \sigma_B^2(th) \) is the Otsu’s variance for a given \( th \) value. Otsu’s method is applied for a single component of an image, that means for RGB images it is necessary to apply separation into single component images. The previous illustration of such bi-level method can be modified for multiple thresholds. The objective function \( F_{otsu}(th) \) in Eq. (27) can also be modified for multiple thresholds as follows:

\[ F_{otsu}(TH) = \max(\sigma_B^2(th)) \text{ where } 0 \leq th \leq L - 1 \text{ and } i = [1, 2, 3, \ldots, k] \]  

(28)

where \( TH = [th_1, th_2, \ldots, th_k - 1] \) is a vector containing multiple thresholds, \( L \) denotes maximum grey level, whereas the variances are computed through Eq. (29).

\[ N\sigma_B^2 = \sum_{i=1}^{k} \sigma_i = \sum_{i=1}^{k} \omega_i(\mu_i - \mu_T)^2 \]  

(29)

where \( i \) represents a specific class, \( \omega_i \) and \( \mu_i \) are the probability of occurrence and the mean of a class respectively. For multi-level thresholding, such values are obtained as:

\[ \omega_{k-1}(th) = \sum_{i=th_{k-1}+1}^{L} Ph_i \]  

(30)

for mean values:

\[ \mu_{k-1} = \sum_{i=th_{k-1}+1}^{L} iPh_i \]  

(31)

\[ F_{kapur}(th) = H_1 + H_2 \]  

(32)

B. OTSU’S OBJECTIVE FUNCTION

Otsu was selected because it is commonly used for thresholding images, segmented by maximizing the between-class variation. TSA–LEO optimizer maximizes the Otsu objective function and determines the best-fit thresholds. The objective function of Otsu considers \( L \) intensity levels of a gray image, and the probability distribution is computed in Eq. 21. This method can be used for RGB color images in which Otsu is separately applied to each channel.
where the entropies $H_1$ and $H_2$ are computed as:

$$H_1 = \sum_{i=1}^{th} \frac{\omega_i}{\omega_0} \ln \left( \frac{\omega_i}{\omega_0} \right)$$

$$H_2 = \sum_{i=th+1}^{L} \frac{\omega_i}{\omega_1} \ln \left( \frac{\omega_i}{\omega_1} \right)$$

(33)

where $\omega_i$ is the probability distribution of the intensity levels which is obtained using Eq. (13), $\omega_0(\theta)$ and $\omega_1(\theta)$ are probabilities distributions for the classes $C_1$ and $C_2$. $\ln(.)$ is the natural logarithm. Similar to the Otsu’s method, the entropy-based approach can be modified for multi-thresholding values; for such a case, it is necessary to divide the image into $k$ classes using $k - 1$ thresholds. The objective function then can be modified as follows:

$$F_{kapur}(TH) = \sum_{i=1}^{k} H_i$$

(34)

where $TH = [th_1, th_2, \ldots, th_{k-1}]$ is a vector that contains the multiple thresholds. Each entropy is computed separately with its respective $(\theta_i)$ value, so Eq. (34) is expanded for $k$ entropies as:

$$H_k^i = \sum_{i=th_{k-1}}^{L} \frac{\omega_i}{\omega_{k-1}} \ln \left( \frac{\omega_i}{\omega_{k-1}} \right)$$

(35)

Here the values of the probability occurrence $(\omega_0^i, \omega_1^i, \ldots, \omega_{k-1}^i)$ of the $k$ classes are obtained using Eq. (20) and the probability distribution $\omega_i$ with Eq. (13).

For the ease of understanding TSA-LEO implementation on image segmentation, the following steps are given in brief.

1) Read the image in grayscale.
2) Obtain the histogram of the selected image.
3) Calculate the probability distribution using Eq23.
4) Initialize TSA–LEO parameters.
5) Initialize the first population of tunicates $P_p$ with the dimension of Dim.
6) Evaluate the initial population using Otsu ($F_{otsu}$) Eq28 or Kapur ($F_{kapur}$) Eq34.
7) Calculate the parameters $\hat{A}$, $\hat{G}$, $\tilde{F}$, $\tilde{M}$, and $\tilde{PD}$ using Eqs.(1-5) respectively.
8) Update the positions of each agent using Eqs.(6 or 7).
9) Determine the optimal position $X_{best}$.
10) Apply LEO strategy if $rand < pr$ and update the value of $P_p$ based on Eq8 if $rand < 0.5$ or 9 if $rand \geq 0.5$.
11) Evaluate the new population and save best results.
12) Select tunicate with the best solution according to the objective function.
13) If maximum iteration or the stop conditions are not met, go to Step 7.
14) To segment the image, use tunicate with the best threshold values.

D. ENVIRONMENTAL SETUP

The results of advanced TSA–LEO with the objective functions of Otsu and Kapur were compared with those of MFO [2], WOA [3], SCA [4], SOA [5], BMO [7], CTSA, and original TSA. All algorithms were executed 35 times per algorithm under the same stopping criteria (350 iterations at most) with 50 search agents to evaluate their performances. The parameters of each algorithm were maintained at their standard versions’ values. All tested algorithms were programmed and operated in the same experimental environment (Intel Core-i5 processor, 8 GB memory, Matlab-2013, and Windows 8.1-64).

E. EVALUATION CRITERIA

Evaluating segmented images is essential for validating the performance and accuracy of any algorithm. Three measures were used to evaluate the degree of segmentation: PSNR [59], SSIM [60], and FSIM [61]. Wilcoxon rank-sum was used to evaluate the significance of the proposed TSA–LEO, and the variations between the proposed method and competing algorithms were assessed in Friedman’s non-parametric statistical tests [62], [63].

1) QUALITY METRICS

The PSNR distinguishes between the qualities of the initial and resulting images. The PSNR is defined as

$$PSNR = 20 \log_{10} \frac{255}{RMSE}$$

RMSE = $\sqrt{\frac{\sum_{i=1}^{M} \sum_{j=1}^{N} ((I(i,j) - Seg(i,j))^2)}{M \times N}}$

(36)

where $RMSE$ is the root-mean-squared error, and $I$ and $Seg$ are the initial and final images, respectively. All images are sized $M \times N$.

The SSIM determines the similarity between the original and segmented images. The SSIM is defined as

$$SSIM(I, Seg) = \frac{(2\mu_I \mu_{Seg} + c_1)(2\sigma_{I,Seg} + c_2)}{(\mu_I^2 + \mu_{Seg}^2 + c_1)(\sigma_I^2 + \sigma_{Seg}^2 + c_2)}$$

(37)

where $\mu_I$ and $\mu_{Seg}$ are the mean intensities of the original image $I$ and segmented image $Seg$, respectively, and $\sigma_I$ and $\sigma_{Seg}$ are their respective standard deviations. $\sigma_{I,Seg}$ is the covariance of the original and segmented images, and $c_1$ and $c_2$ are two constants.

The FSIM measures the similarities in the mapped features. The FSIM mainly depends on the phase congruency (PC) and gradient magnitude (GM). The PC is a new measure applied to the features of an image. The GM computes the image gradient, as traditionally done in digital image processing. The similarity between the two images was first obtained as

$$S_{PC} = \frac{2PC_1PC_2 + T_1}{PC_1^2 + PC_2^2 + T_1}$$

(38)

where $T_1$ is a positive constant that increases the stability of $S_{PC}$. $PC_1$ and $PC_2$ are the PCs of the original and segmented images, respectively, and $SG$ is the similarity between $G_1$ and $G_2$, which is computed as:

$$S_G = \frac{2G_1G_2 + T_2}{G_1^2 + G_2^2 + T_2}$$

(39)
Here, $G_1$ and $G_2$ are the gradients of the original and segmented images, respectively, and $T_2$ is a positive constant that depends on the dynamic range of GM values. From Eqs. (38) and (39), the similarity is computed as

$$S_L(x) = [S_{PC}(x)]^\alpha [S_{C}(x)]^\beta$$  \hspace{1cm} (40)

The parameters $\alpha$ and $\beta$ adjust the relative importances of the PC and GM features. Note that high values of fitness, PSNR, SSIM, and FSIM indicate a high-performing algorithm.

2) NON-PARAMETRIC STATISTICAL TESTS

The Wilcoxon rank-sum test is a non-parametric measure that analyzes the results of pairs of methods. The null hypothesis implies that the ranks of the results of the comparative methods are not significantly different. The alternative hypothesis examines whether the results of the comparative methods can be distinguished by rank. The Wilcoxon rank-sum was calculated at the 5% significance level. The significance levels (P) and hypothesis (H) values in terms of fitness obtained with Otsu’s method are shown in Table 19. If P > 0.05 or H = 0, then the null hypothesis is accepted, whereas if P < 0.05 or H = 1, the alternative hypothesis is accepted.

The Friedman mean rank test is another non-parametric analysis that compares three or more matched groups. In the present study, the Friedman mean rank was applied for checking the performances of the competitive algorithms. The Friedman statistic determines the mean ranked value. Whether the critical values reach the assigned significance level is evaluated using Friedman’s statistic, and whether the null hypothesis is accepted or declined is then judged.

F. ANALYSIS OF MULTI-THRESHOLDING IMAGE SEGMENTATION RESULTS

This section reports and discusses the experimental results of multilevel thresholding Otsu and Kapur objective functions described above to tackle multilevel thresholding image-segmentation problems.

1) MULTI-THRESHOLDING SEGMENTATION EXPERIMENTS OF OTSU AND KAPUR METHODS IN TABLES AND FIGURES

Image-segmentation experiments were performed using Otsu and Kapur methods as the objective functions in two separate experiments on a set of ten benchmark images at four thresholding levels (Level = 2, 3, 4, and 5). In total, 40 cases were tested. Fig. 7 illustrates a set of benchmark images with their respective histograms namely Cameraman, Lena, Baboon, Hunter, Airplane, Pepper, Living room, Woman, Bridge, and Butter-Fly. Figures (12, 13, TSA-LEO Otsu) and (12, 13, TSA-LEO Kapur) show the segmented image results with their respective selected thresholds over histograms obtained from applying the proposed TSA-LEO with Otsu and Kapur methods. In addition, Tables (14, TSA-LEO OTSU) and (6, TSA-LEO Kapur) show the optimal thresholds obtained from TSA-LEO and other competitors under the condition of level = 2, 3, 4, and 5 for Otsu and Kapur objective functions. Tables (15, 16, 17, 18, TSA-LEO Otsu) and (7, 8, 9, and 10, TSA-LEO Kapur) represent the fitness, PSNR, SSIM, and FSIM results, for Otsu and Kapur methods respectively. Moreover, Tables (19 and 19) show the Wilcoxon rank-sum test of TSA–LEO and other seven algorithms with Otsu and Kapur methods. Table 20 also provides convergence curves on Otsu and Kapur objective functions for samples of test images on various thresholds for the proposed TSA-LEO and other competitive algorithm.

2) MULTI-THRESHOLDING SEGMENTATION ANALYSIS OF OTSU AND KAPUR METHODS

From the optimal thresholds selected on the basis of Otsu and Kapur objective functions, we can conclude that the Kapur segmentation process is more decentralized and has wider coverage, such as the optimal threshold value of the test image, namely, Cameraman at Level = 4 is 22, 59, 98, 145, 196, as shown in Table 6, and 36, 82, 122, 149, 173 shown in Table 14. Notably, the optimal thresholds for Otsu’s objective function are closer than Kapur’s objective function, revealing that segmentation based on Kapur’s objective function is better than Otsu’s objective function, which is also evident from the results of the segmented images based on Otsu and Kapur objective functions in Figures (12, 13, 4, and 5). In terms of quality metrics (PSNR, SSIM, and FSIM), the quality of the segmented image based on Kapur’s is better than Otsu’s objective function. For example, the PSNR value of starfish in Test 9 image at Level = 4 is 1.90E + 01 in Table 8 and 1.89E + 01 in Table 16, SSIM value is 8.70E − 01 8 and 8.46E − 01 16, FSIM value is 9.00E − 01 in Table 8 and 8.86E − 01 16. The values of the Otsu-based method are smaller than those of Kapur’s entropy-based method, and the segmentation effect in Tables 4, 5 is clearer than in Tables 12, 13, especially in the case with higher number of thresholds. Generally, in the given segmentation image and the same number of threshold levels, the method based on Kapur’s is significantly better than Otsu-based method for the same optimization algorithm.
TABLE 6. Optimal thresholds obtained by Kapur’s objective function.

| Test Image | Level | MFO | WOA | SCA | BPSO | BO | TSA | CTSA | TSA–LEO |
|------------|-------|-----|-----|-----|-----|----|-----|------|--------|
| Test 1     | 2     | 138 | 138 | 138 | 127 | 127 | 125 | 125  | 125    |
| Test 2     | 3     | 125 | 125 | 125 | 125 | 125 | 125 | 125  | 125    |

TABLE 7. Average and STD of Kapur’s fitness obtained from all algorithms.

| Test Image | Level | MFO | WOA | SCA | BPSO | BO | TSA | CTSA | TSA–LEO |
|------------|-------|-----|-----|-----|-----|----|-----|------|--------|
| Test 1     | 2     | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768   |
| Test 2     | 3     | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768 | 1.768   |

3) MULTI-THRESHOLDING SEGMENTATION ANALYSIS OF TSA-LEO AND OTHER SEVEN OPTIMIZATION ALGORITHMS

Fitness results in the basis of Kapur’s objective function shown in Table 7 confirmed the superiority of TSA–LEO over other algorithms. TSA–LEO ranked first with 21 higher fitness cases (52.5%); MFO and WOA ranked second with 8 higher fitness cases (20%); and CTSA ranked third with 7 higher fitness cases (17.5%). Moreover, BMO ranked fourth with only 4 higher fitness cases (10%). All the remaining algorithms have no higher fitness cases. Regarding PSNR results...
shown in Table 8, the proposed TSA-LEO ranked first with 19 higher cases (47.5%) and CTSA ranked second with 7 higher cases (17.5%). Besides, SOA ranked third with 6 higher cases (15%). WOA ranked fourth with 5 higher best fitness cases (12.5%). MFO and BMO ranked fifth with 3 higher cases representing (7.5%) of overall higher cases. Finally, TSA took the last rank without any higher case.

The proposed TSA-LEO ranked first in the SSIM results represented in Table 9, with 21 higher cases representing (52.5%) of overall test cases. However, SOA ranked second with 7 higher cases (17.5%). BMO gained third place with only 5 higher cases representing (12.5%) from overall cases. VOLUME 9, 2021
TABLE 10. Average and STD of FSIM based Kapur’s objective function obtained from all algorithms.

| Test Image | MFO | WOA | SCA | SOA | BMO | TSA | CTSA |
|------------|-----|-----|-----|-----|-----|-----|-----|
|             |     |     |     |     |     |     |     |

TABLE 11. Comparison of the p-values obtained from the Wilcoxon signed-rank test between the pairs of TSA–LEO vs. MFO, TSA–LEO vs. WOA, TSA–LEO vs. SCA, TSA–LEO vs. SOA, TSA–LEO vs. BMO, TSA–LEO vs. TSA, and TSA–LEO vs. CTSA for Kapur’s method in terms of Fitness results.

MFO, WOA, SCA, and CTSA ranked fourth with 3 higher FSIM cases representing a (7.5%) of the total cases. Finally, TSA ranked last without any higher cases. Regarding the FSIM results shown in Table 10, proposed TSA-LEO ranked first with 18 higher cases (45%). MFO was ranked second with 7 higher cases (17.5%) and WOA ranked third with 5 higher cases (12.5%). CTSA ranked fourth with 4 higher cases (10.5%). SCA and BMO ranked fifth with only one higher case with a percentage of (2.5%). In addition, TSA gained no higher FSIM cases.
TABLE 12. Segmented images after applying TSA–LEO on Otsu’s method.

| Image  | level = 2 | level = 3 | level = 4 | level = 5 |
|--------|-----------|-----------|-----------|-----------|
| Cameraman | ![Segmented Image](image1.png) | ![Segmented Image](image2.png) | ![Segmented Image](image3.png) | ![Segmented Image](image4.png) |
| Lena | ![Segmented Image](image5.png) | ![Segmented Image](image6.png) | ![Segmented Image](image7.png) | ![Segmented Image](image8.png) |
| Baboon | ![Segmented Image](image9.png) | ![Segmented Image](image10.png) | ![Segmented Image](image11.png) | ![Segmented Image](image12.png) |
| Hunter | ![Segmented Image](image13.png) | ![Segmented Image](image14.png) | ![Segmented Image](image15.png) | ![Segmented Image](image16.png) |
| Airplane | ![Segmented Image](image17.png) | ![Segmented Image](image18.png) | ![Segmented Image](image19.png) | ![Segmented Image](image20.png) |

Regarding the objective function of Otsu, the results of Otsu in terms of the mean of fitness provided in Table 15 confirmed the superiority of TSA–LEO over the other algorithms. Where the proposed TSA–LEO ranked first with 40 higher cases (100%), Table 16 shows the mean PSNR results and confirms that BMO ranked first with 10 higher cases (25%). Moreover, the proposed TSA–LEO and SCA ranked second with 9 higher cases (22.5%). MFO ranked third with 8 higher cases (20%) and CTSA ranked fourth with 4 higher best fitness cases (10%). The original TSA and SOA with 3 higher cases represent (7.5%) of overall higher cases. Finally, WOA ranked in the last place with only one higher case representing (2.5%). Table 17 represents the SSIM results of the proposed TSA–LEO as compared with other algorithms in terms of SSIM mean results. Remarkably, BMO ranked first with 15 higher cases representing (37.5%) of overall test cases. TSA–LEO ranked second with 9 higher cases (22.5%). SCA ranked third with overall 5 higher cases (12.5%). Moreover, WOA, SOA, CTSA ranked fourth with 4 higher cases representing a percentage of (10%). Besides, TSA gained two higher FSIM cases with (5%) of overall cases. Finally, MFO ranked last with only one higher FSIM case representing (2.5%) of overall cases. According to the Wilcoxon rank sum test, Tables (11 and 19) represent P and H results of the Wilcoxon test in terms of fitness for Kapur and Otsu objective functions, respectively. When the number of thresholds is small (e.g., Level = 2, 3), the segmentation results of each algorithm are almost the same, according to comparisons based on Kapur and For example, when Level = 2, the optimal threshold, PSNR, SSIM, and FSIM of the eight algorithms of Baboon are the same.
When the number of thresholds is large (e.g., Level = 4, 5), the numerical difference of segmentation results is obvious.

For example, the PSNR values of image Test 4 at Level = 5 are different: 21.78 Table (8, MFO), 20.99 Table (8, WOA), 56.086

VOLUME 9, 2021
TABLE 16. Average and STD of PSNR based Otsu’s objective function obtained from all algorithms.

| Test Image | MFO | WOA | SCA | BMO | CSA | TSA | CSA-LEO |
|------------|-----|-----|-----|-----|-----|-----|---------|
| Level 1     |     |     |     |     |     |     |         |
| Level 2     |     |     |     |     |     |     |         |
| Level 3     |     |     |     |     |     |     |         |
| Level 4     |     |     |     |     |     |     |         |
| Level 5     |     |     |     |     |     |     |         |

TABLE 17. Average and STD of SSIM based Otsu’s objective function obtained from all algorithms.

| Test Image | MFO | WOA | SCA | BMO | CSA | TSA | CSA-LEO |
|------------|-----|-----|-----|-----|-----|-----|---------|
| Level 1     |     |     |     |     |     |     |         |
| Level 2     |     |     |     |     |     |     |         |
| Level 3     |     |     |     |     |     |     |         |
| Level 4     |     |     |     |     |     |     |         |
| Level 5     |     |     |     |     |     |     |         |

E. H. Houssein et al.: Improved TSA for Global Optimization and Image Segmentation

21.71 Table (8, SCA), 22.22 Table (8, SOA), 21.05 Table (8, BMO), 17.94 Table (8, TSA), 22.48 Table (8, CTSA), and 3.08E+01 Table (8, TSA-LEO); SSIM and FSIM also have similar differences. In Kapur’s method, it is noticeable that the value based on TSA-LEO occupies the most advantages. However, results based on the Otsu method are close to that of VOLUME 9, 2021 56087
Kapur’s method but ranked second in terms of fitness and all other quality measures. Generally, the segmentation effect of TSA-LEO reflects the superiority of the proposed TSA-LEO especially in Kapur’s method.

### TABLE 18. Average and STD of FSIM based Otsu’s objective function obtained from all algorithms.

| Test Image | Level | MFO | WOA | SOA | TSA | CTSA | BMO | MFO-LEO | WOA-LEO | SOA-LEO | TSA-LEO | CTSA-LEO | BMO-LEO |
|------------|-------|-----|-----|-----|-----|------|-----|--------|---------|---------|--------|--------|--------|
| 1          | 1     | 3.59 | 3.72 | 3.74 | 3.66 | 3.73 | 3.59 | 3.72   | 3.74    | 3.66    | 3.73   | 3.59   | 3.72   |
| 2          | 1     | 3.60 | 3.72 | 3.74 | 3.66 | 3.73 | 3.59 | 3.72   | 3.74    | 3.66    | 3.73   | 3.59   | 3.72   |

### TABLE 19. Comparison of the p-values obtained from the Wilcoxon signed-rank test between the pairs of TSA–LEO vs. MFO, TSA–LEO vs. WOA, TSA–LEO vs. SCA, TSA–LEO vs. SOA, TSA–LEO vs. BMO, TSA–LEO vs. TSA, and TSA–LEO vs. CTSA for Otsu’s method in terms of Fitness results.

| Test Image | Level | MFO | WOA | SOA | TSA | CTSA | BMO | MFO-LEO | WOA-LEO | SOA-LEO | TSA-LEO | CTSA-LEO | BMO-LEO |
|------------|-------|-----|-----|-----|-----|------|-----|--------|---------|---------|--------|--------|--------|
| 1          | 1     | 3.59 | 3.72 | 3.74 | 3.66 | 3.73 | 3.59 | 3.72   | 3.74    | 3.66    | 3.73   | 3.59   | 3.72   |
TABLE 20. The convergence curves for the proposed I–EO and the competitor algorithms for multi-threshold image segmentation problems.

![Image showing convergence curves for different test cases.]

VI. CONCLUSION AND FUTURE WORK
This paper introduced an enhanced variant of a metaheuristic optimization algorithm, named TSA. The TSA was hybridized with an efficient search strategy called LEO, which improves the performance, accuracy, and convergence behavior of TSA. During the solution update process, TSA competes with LEO in the proposed TSA–LEO method. The effectiveness of the proposed TSA–LEO was...
evaluated using the functions in the CEC’17 benchmark test suite. The proposed method outperformed the competing methods regarding various statistical measures. Moreover, the proposed TSA–LEO can tackle multilevel threshold problems while seeking the optimal thresholds for image separation. Thus, the proposed TSA–LEO method is potentially applicable for solving complicated real-world problems. The proposed method selects the optimal thresholds that intensified the segmentation process in the thresholding experiment.

In future work, we intend to 1) combine two or more objective functions (e.g., Otsu and Kapur) in the proposed TSA–LEO, 2) further evaluate the proposed method on different datasets, and 3) apply the proposed TSA–LEO to other real-world complex problems. Promisingly, the proposed approach can be considered as an efficient and effective strategy for more complex optimization scenarios and the intelligent optimization field’s theoretical work as well.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

CREDIT AUTHOR STATEMENT

Essam H. Houssein: Supervision, Methodology, Conceptualization, Software, Formal analysis, Writing - review & editing. Bahaa El-din Helmy: Software, Resources, Writing - original draft. Ahmed A. Elngar: Conceptualization, Formal analysis, Writing - review & editing. Diaa Salama AbdElmnaam: Methodology, Conceptualization, Formal analysis, Writing - review & editing. Hassan Shaban: Conceptualization, Formal analysis, Writing - review & editing. All authors read and approved the final paper.

REFERENCES

[1] X.-S. Yang, “Nature-inspired optimization algorithms: Challenges and open problems,” J. Comput. Sci., vol. 46, Oct. 2020, Art. no. 101104.
[2] S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” Knowl.-Based Syst., vol. 89, pp. 228–249, Nov. 2015.
[3] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” Adv. Eng. Softw., vol. 95, pp. 51–67, May 2016.
[4] S. Mirjalili, “SCA: A sine cosine algorithm for solving optimization problems,” Knowl.-Based Syst., vol. 96, pp. 120–133, Mar. 2016.
[5] G. Dhiman and V. Kumar, “Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems,” Knowl.-Based Syst., vol. 165, pp. 169–196, Feb. 2019.
[6] L. M. Abualigah, A. T. Khader, E. S. Hanande, and A. H. Gandomi, “A novel hybridization strategy for krill herd algorithm applied to clustering techniques,” Appl. Soft Comput., vol. 60, pp. 423–435, Nov. 2017.
[7] A. H. Sulaiman, Z. Mustaffa, M. M. Saari, and H. Daniyal, “Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems,” Eng. Appl. Artif. Intell., vol. 87, Jan. 2020, Art. no. 103330.
[8] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67–82, Apr. 1997.
[9] A. Mostafa, E. H. Houssein, M. Houseni, A. E. Hassanien, and H. Hefny, “Evaluating swarm optimization algorithms for segmentation of liver images,” in Advances in Soft Computing and Machine Learning in Image Processing, vol. 730. Cham, Switzerland: Springer, 2018, pp. 41–62.
[10] S. Said, A. Mostafa, E. H. Houssein, A. E. Hassanien, and H. Hefny, “Moth-flame optimization based segmentation for mri liver images,” in Proc. Int. Conf. Adv. Intell. Syst. Inform., vol. 639. Cham, Switzerland: Springer, 2017, pp. 320–330.
[11] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany, “A modified henry gas solubility optimization for solving motif discovery problem,” Neural Comput. Appl., vol. 32, no. 14, pp. 10759–10771, Jul. 2020.
[12] A. E. Hassanien, M. Kilany, E. H. Houssein, and H. Alqheeri, “Intelligent human emotion recognition based on elephant herding optimization tuned support vector regression,” Biomed. Signal Process. Control, vol. 45, pp. 182–191, Aug. 2018.
[13] E. H. Houssein, M. E. Hosney, D. Oliva, W. M. Mohamed, and M. Hassaballah, “A novel hybrid harrs hawks optimization and support vector machines for drug design and discovery,” Comput. Chem. Eng., vol. 133, Feb. 2020, Art. no. 106656.
[14] E. H. Houssein, M. E. Hosney, E. H. Houssein, M. H. Elhosseny, D. Oliva, W. M. Mohamed, and M. Hassaballah, “Hybrid harrs hawks optimization with cuckoo search for drug design and discovery in chemoinformatics,” Sci. Rep., vol. 10, no. 1, Dec. 2020, Art. no. 14439.
[15] N. Neggaz, E. H. Houssein, and K. Hussain, “An efficient henry gas solubility optimization for feature selection,” Expert Syst. Appl., vol. 152, Aug. 2020, Art. no. 113646.
[16] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, “Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems,” Int. J. Speech Technol., vol. 51, no. 3, pp. 1531–1551, Mar. 2021.
[17] F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, and S. Mirjalili, “Henry gas solubility optimization: A novel physics-based algorithm,” Future Gener. Comput. Syst., vol. 101, pp. 664–667, Dec. 2019.
[18] E. H. Houssein, M. R. Saad, F. A. Hashim, H. Shaban, and M. Hassaballah, “Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems,” Eng. Appl. Artif. Intell., vol. 94, Sep. 2020, Art. no. 103731.
[19] D. A. Oliva Navarro, S. Hinojosa, and M. Demeshko, “Engineering applications of metaheuristics: An introduction,” I. Phys., Conf. Ser., Inf. Technol. Bus. Ind., vol. 8032016, Oct. 2017, Art. no. 12111.
[20] L. M. Q. Abualigah, Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering, vol. 816. Cham, Switzerland: Springer, 2019.
[21] L. M. Abualigah, A. T. Khader, and E. S. Hanande, “Hybrid clustering analysis using improved krill herd algorithm,” Int. J. Speech Technol., vol. 48, no. 11, pp. 4047–4071, Nov. 2018.
[22] M. M. Ahmed, E. H. Houssein, A. E. Hassanien, A. Taha, and E. Hassanien, “Maximizing lifetime of large-scale wireless sensor networks using multi-objective whale optimization algorithm,” Telecommun. Syst., vol. 72, no. 2, pp. 243–259, Oct. 2019.
[23] E. H. Houssein, M. R. Saad, K. Hussain, W. Zhu, H. Shaban, and M. Hassaballah, “Optimal sink node placement in large scale wireless sensor networks based on ‘Herr’ hawks optimization algorithm,” IEEE Access, vol. 8, pp. 19381–19397, 2020.
[24] M. Črepinšek, S.-H. Liu, and M. Mernik, “Exploration and exploitation in evolutionary algorithms: A survey,” ACM Comput. Surv., vol. 45, no. 3, pp. 1–33, Jun. 2013.
[25] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, “A better balance in Metaheuristic algorithms: Does it exist?” Swarm Evol. Comput., vol. 72, May 2020, Art. no. 100671.
[26] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, “An improved grey wolf optimizer for solving engineering problems,” Expert Syst. Appl., vol. 166, Mar. 2021, Art. no. 113917.
[27] J. Luo, H. Chen, A. A. Heidari, Y. Xu, Q. Zhang, and C. Li, “Multi-strategy boosted mutative whale-inspired optimization approaches,” Appl. Math. Model., vol. 73, pp. 109–123, Sep. 2019.
[28] M. Issa, A. E. Hassanien, D. Oliva, A. Helmi, I. Ziedan, and A. Alzohairy, “ASCA-PSO: Adaptive sine cosine optimization algorithm integrated with particle swarm for pairwise local sequence alignment,” Expert Syst. Appl., vol. 99, pp. 56–70, Jun. 2018.
[29] M. Abd Elaziz, D. Oliva, and S. Xiong, “An improved opposition-based sine cosine algorithm for global optimization,” Expert Syst. Appl., vol. 90, pp. 484–500, Dec. 2018.
[30] H. Zhang, A. A. Heidari, M. Wang, L. Zhang, H. Chen, and C. Li, “Orthogonal nelder-mead moth flame method for parameters identification of photovoltaic modules,” Energy Convers. Manage., vol. 211, May 2020, Art. no. 112764.
[31] S.-H. Kim, K.-J. An, S.-W. Jing, and G.-Y. Kim, “Texture feature-based text region segmentation in social multimedia data,” Multimedia Tools Appl., vol. 75, no. 20, pp. 12815–12829, Oct. 2016.
AHMED A. ELNGAR is currently an Assistant Professor with the Department of Computer Science, Faculty of Computers and Artificial Intelligence, Beni-Suef University. His current project is biometrics, AI-based security, image processing, AI-based control, AI-based smart grids, and the Internet of Things Security. He is also the Founder and Chair of the Scientific Innovation Research Group (SIRG). He is also a Managing Editor of Journal of Cybersecurity and Information Management (JCIM).

DIAA SALAMA ABDELMINAAM received the Ph.D. degree in information system from the Faculty of Computers and Information, Menofia University, Egypt, in 2015. Since 2011, he has been an Assistance Professor with the Department of Information Systems, Faculty of Computers and Information, Benha University, Egypt. He has worked on several research topics. He has contributed more than 40 technical papers in the areas of wireless networks, wireless network security, information security and Internet applications, cloud computing, mobile cloud computing, the Internet of Things, and machine learning in international journals, international conferences, local journals, and local conferences. He majors in cryptography, network security, the IoT, big data, cloud computing, and deep learning.

HASSAN SHABAN received the Ph.D. degree in computer science. He currently works as a Lecturer with the Department of Computer Science, Faculty of Computers and Information, Minia University, Egypt. His research interests include wireless sensor networks, security, optimization, and metaheuristics.