An Enhanced Secretary Bird Optimization Algorithm for Image Segmentation using Multi-level Thresholding Strategy Applied on New Dermatology Medical Dataset

Ruba Abu Khurma[a,b] (ruba_abukhurma@yahoo.com), Marwa M. Emam[c] (marwa.khalef@mu.edu.eg), Falguni Chakraborty[d] (falguni.durgapur@gmail.com), Saher Mohamed[d] (sahermuhamed176@gmail.com)

[a] MEU Research Unit, Faculty of Information Technology (Middle East University) Amman 11831, Jordan
[b] and Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan
[c] Faculty of Computers and Information, Minia University, Minia, Egypt.
[c] Department of Computer Science & Engineering, National Institute of Technology, Durgapur, West Bengal, India. [d]Faculty of Computers and Artificial Intelligence, Benha University,Benha, Al Qalyubiyah, Egypt.

Corresponding Author:
Last Author
Full address of the corresponding author, including the country name
Tel: (555) 555-1234
Email: last.author@mail.com

An Enhanced Secretary Bird Optimization Algorithm for Image Segmentation using Multi-level Thresholding Strategy Applied on New Dermatology Medical Dataset

First Author[a,b], Second Author[a], Last Author[c,*]

[a]*Full address of first author, including the country name*
[b]*Full address of second author, including the country name*
[c]*Full address of last author, including the country name*

Abstract

A

Keywords: Secretary Bird Optimization Algorithm (SBOA), Orthogonal learning (OL), Opposition Learning (OL), Image segmentation, Multi-level Thresholding, SCINE dataset

1. Introduction

1.1. Main topic and scope

1.2. Brief history of image segmentation

1.3. Niche in previous studies

1.4. Relevance

1.5. Motivation

2. A thorough literature review about adopting NIA for Image segmentation

2.1. Image segmentation problem: background

2.2. Adopting NIA for image segmentation

2.3. Gaps and unsolved matters

3. Methodology

3.1. Secretary Bird Optimization Algorithm (SBOA)

3.2. Enhancement techniques for SBOA

3.2.1. Opposition-Based Learning

Opposition-based learning (OBL) (Tizhoosh, 2005) is a valuable technique for preventing stagnation in competitive solutions (Aarts et al., 2003). This concept enhances the exploitation aspect of the search mechanism. In meta-heuristic algorithms, convergence tends to occur quickly if the initial solutions are near the optimal position; otherwise, convergence is slower. OBL improves outcomes by exploring search regions close to the global optimum. It operates by simultaneously searching in two directions: one defined by the current solution and the other by its opposite. The OBL strategy selects the best direction from all solutions (Houssein et al., 2022).

*Corresponding author.

Email addresses: first.author@mail.com (First Author), second.author@mail.com (Second Author), last.author@mail.com (Last Author)

- **Opposite Number:** The notion of opposite numbers is fundamental to OBL. For a real number $Y_0 \in [u, p]$, the opposite number is given by Eq. (1) (Emam et al., 2023):

$$\overline{\mathrm{Y}}_0 = u + p - Y_0. \tag{1}$$

In an M-dimensional space, the opposite number is calculated using Eq. (2) and Eq. 3:

$$Y = [Y_1, Y_2, Y_3, \ldots, Y_M] \tag{2}$$

$$\overline{\mathrm{Y}} = [\overline{\mathrm{Y}}_1, \overline{\mathrm{Y}}_2, \overline{\mathrm{Y}}_3, \ldots, \overline{\mathrm{Y}}_M] \tag{3}$$

Each component of $\overline{\mathrm{Y}}$ is defined by Eq. (4):

$$\overline{\mathrm{Y}}_k = u_q + p_q - Y_q \quad \text{where} \quad q = 1, 2, 3, \ldots, M \tag{4}$$

- **Opposition-Based Optimization:** In this optimization approach, the opposite value $\overline{\mathrm{Y}}_0$ is evaluated against the current solution Y_0 using the fitness function. If $f_t(Y_0)$ is superior to $f_t(\overline{\mathrm{Y}}_0)$, then Y_0 remains unchanged; otherwise, Y_0 is replaced by $\overline{\mathrm{Y}}_0$. Consequently, the solutions are updated to reflect the best value between Y and $\overline{\mathrm{Y}}$ (Emam et al., 2023).

3.2.2. Orthogonal Learning (OL)

Orthogonal Learning (OL) is a technique extensively used to enhance the search for optimal solutions, striking a balance between exploration and exploitation phases (Zhang et al., 2020). The orthogonal experimental design (OED) method is utilized to develop an OL strategy, crafting efficient agents that guide the population towards the global optimal solution (Gao et al., 2013). The OED determines the best combination of factor levels using a small sample of experiments, which provides new solutions that steer the search in the optimal direction. The OL strategy is implemented in two primary stages:

Orthogonal Array (OA): The first stage involves generating a predefined table known as the orthogonal array (OA). This table consists of a series of distinct numbers typically represented as $L_M(L^Q)$, where $M = 2 \cdot \log_2(D + 1)$. The generated OA has the following properties: L levels per factor and Q factors. For instance, the OA for a three-dimensional Sphere function $O_{array}(2^3)$ is illustrated below.

$$\begin{pmatrix} Combination & x1(Level) & x2(Level) & x3(Level) & Fitness \\ C1 & 1(1) & 2(1) & 3(1) & f(C1) = 14 \\ C2 & 1(1) & 1(2) & 4(2) & f(C2) = 18 \\ C3 & 2(2) & 2(1) & 4(2) & f(C3) = 24 \\ C4 & 2(2) & 1(2) & 3(1) & f(C4) = 14 \end{pmatrix}$$

In this example, the OA comprises three columns, indicating its application for problems with up to three factors, each having two levels.

Factor Analysis (FA): The second stage involves factor analysis, which evaluates the effect of each level on each factor based on the experimental results of all M combinations of the OA. This effect is calculated using the following equation:

$$W_{q,l} = \sum_{m=1}^{M} f(C_m) \cdot E_{m,q,l} \tag{5}$$

Here, $W_{q,l}$ represents the effect of the l^{th} level on the q^{th} factor, where $m = 1, 2, 3, \ldots, M$, $q = 1, 2, 3, \ldots, Q$, and $l = 1, 2, 3, \ldots, L$. The term $f(C_m)$ denotes the fitness of the m^{th} combination in the OA. The variable $E_{m,q,l}$ is set to 1 if the l^{th} level is used for the q^{th} factor in the m^{th} combination, and 0 otherwise. Using Eq. (5), the impact of each level on each factor, as shown in Table 1, is quickly determined. By comparing these effects, the optimal combination of levels is identified.

Table 1
Orthogonal experimental design for a three-dimensional Sphere function.

Level	**Factor Analysis**			
L1	f(C1)+f(C2)=32	f(C1)+f(C3)=38	f(C1)+f(C4)=28	
L2	f(C3)+f(C4)=38	f(C2)+f(C4)=32	f(C2)+f(C3)=42	
Best Level	**x1(1)**	**x2(2)**	**x3(1)**	
OED Result	**1**	**1**	**3**	f_{min}**=11**

Overall, the OL strategy can be expressed as Eq. (6):

$$X_n^m = X_{n_{best}}^m \oplus X_n^m \tag{6}$$

In this equation, the symbol $\oplus$ represents the OL operation. In this study, an agent X_n^m is selected from the population, and the best-scoring agent $X_{n_{best}}^m$ is combined with it to generate a new, efficient solution.

3.3. Enhanced Secretary Bird Optimization Algorithm (ESBOA)

3.4. ESBOA based multilevel thresholding image segmenting

3.4.1. Pseudocode

3.4.2. flowchart

3.5. Extensive analysis and preprocessing of SCINE dataset

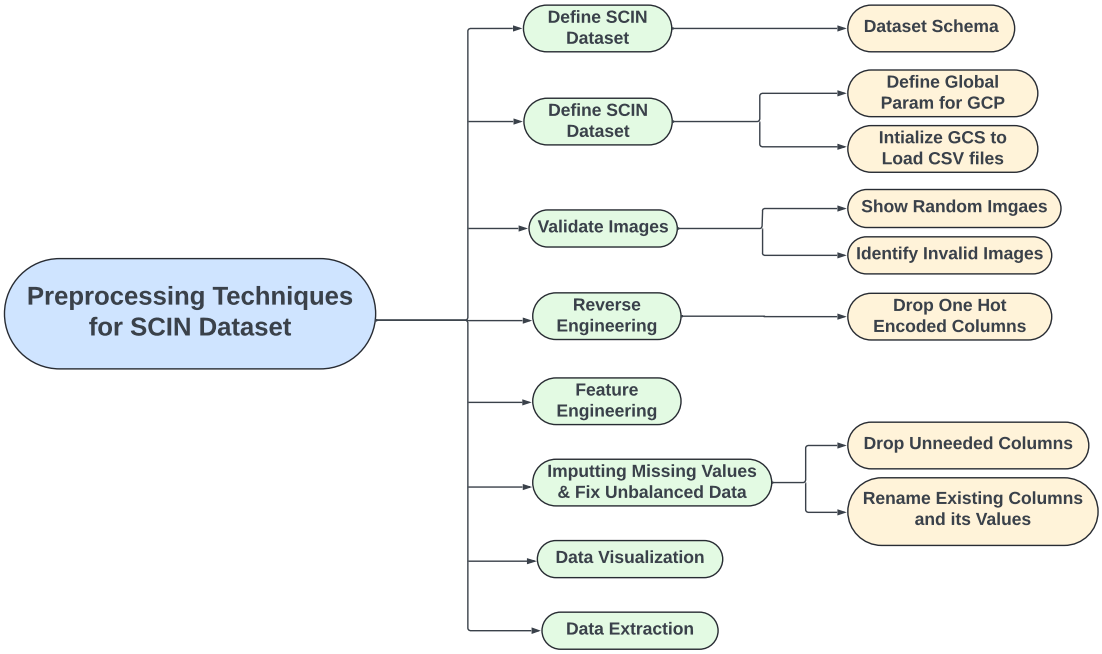
The Skin Condition Image Network (SCIN) dataset offers a diverse and representative collection of skin condition images, bridging important gaps for AI development, medical research, and equitable healthcare tools. Various new techniques in data science are implemented on the unstructured SCIN dataset. Figure 1 shows these techniques:

3.5.1. Defining SCIN dataset

- **Dataset composition:**

  The SCIN dataset currently contains over 10,000 images of skin, nail, or hair conditions, directly contributed by individuals experiencing them. All contributions were made voluntarily with informed consent by individuals in the US, under an institutional-review board-approved study. To provide context for retrospective dermatologist labeling, contributors were asked to take images both close-up and from slightly further away. They were given the option to self-report demographic information and tanning propensity (self-reported Fitzpatrick Skin Type, i.e., sFST), and to describe the texture, duration, and symptoms related to their concern. One to three dermatologists labeled each contribution with up to five dermatology conditions, along with a confidence score for each label. The SCIN dataset contains these individual labels, as well as an

Figure 1 Data science techniques applied on SCIN dataset in preprocessing stage



aggregated and weighted differential diagnosis derived from them that could be useful for model testing or training. These labels were assigned retrospectively and are not equivalent to a clinical diagnosis, but they allow us to compare the distribution of dermatology conditions in the SCIN dataset with existing datasets.

- **Fitzpatrick scale**

  Before work with SCIN dataset you need to know what is Fitzpatrick scale? The Fitzpatrick scale, also known as the Fitzpatrick skin type classification, is a system for classifying human skin color based on the skin's response to ultraviolet (UV) radiation exposure. Developed by dermatologist Thomas B. Fitzpatrick in 1975, this scale helps to categorize skin types and predict the risk of sunburn and skin cancer. It is widely used in dermatology and skincare to tailor treatments and preventive measures based on skin type. preventive measures based on skin type. To know more about Fitzpatrick scale you can click HERE

- **Monk Skin Tone Scale**

  The Monk Skin Tone Scale, also known as the Monk skin type classification, is a system developed for classifying human skin tones. It aims to provide a simple and practical way to categorize skin types based on their susceptibility to sunburn and skin cancer, similar to the Fitzpatrick scale. The Monk Skin Tone Scale typically includes a range of skin types, often categorized into several groups based on their characteristics, such as Light Skin Tones, Medium Skin Tone, Dark Skin

Tones, and Very Dark Skin Tones. To know more about monk skin tone scale and why it is related to Fitzpatrick you can click HERE

3.5.2. Dataset schema

- **Dataset on Google Cloud Storage**

  The data is stored in Google Cloud Storage and to access it click HERE. Check out the load notebook from HERE for a quick review of how to access the dataset and the Dataset Documentation for an overview of its schema from HERE. Please note: This dataset contains images of medical conditions, some of which may be sensitive and/or graphic in nature.

- **Dataset Description**

  1- scin_cases.csv

  - case_id: Identifier for the case.
  - source: Value is "SCIN".
  - release: Identifier for the release of the case, formatted as "major.minor.patch".
  - year: Year closest to the bulk of the data released.
  - age_group: Mapped age ranges: AGE_18_TO_29, AGE_30_TO_39, etc.
  - sex_at_birth: User-reported sex at birth: FEMALE, MALE, OTHER_OR_UNSPECIFIED.
  - fitzpatrick_skin_type: User-reported skin type: FST1-6, NONE_SELECTED.
  - race_ethnicity_*: User-reported race and/or ethnicity demographic.
  - textures_*: User-reported skin condition textures.
  - body_parts_*: User-reported affected body parts.
  - condition_symptoms_*: User-reported symptoms related to the skin condition.
  - other_symptoms_*: User-reported additional symptoms.
  - related_category: User-reported related categories.
  - condition_duration: User-reported duration of the skin condition.
  - image_.path: Path to the image storage location.
  - image_.shot_type: Enum indicating image shot type.

  2-scin_labels.csv

  - case_id: Same as scin_cases.csv.
  - dermatologist_gradable_for_skin_condition: Label indicating if skin condition can be determined.
  - dermatologist_skin_condition_label_name: List of condition names derived from dermatologist labels.
  - dermatologist_skin_condition_confidence: Confidence scores for dermatologist labels.
  - weighted_skin_condition_label: Final differential label generated from dermatologist labels.
  - dermatologist_gradable_for_fitzpatrick_skin_type: Label indicating if Fitzpatrick skin type can be estimated.

- dermatologist_fitzpatrick_skin_type_label: Dermatologist's estimated Fitzpatrick skin type.
- gradable_for_monk_skin_tone_india: Label indicating if Monk skin tone label can be determined in India.
- gradable_for_monk_skin_tone_us: Label indicating if Monk skin tone label can be determined in the US.
- monk_skin_tone_label_india: Monk skin tone label value in India.
- monk_skin_tone_label_us: Monk skin tone label value in the US.

3.5.3. Accessing the Dataset from Google Cloud Storage (GCS)

- **Defines Global Parameters for Google Cloud Platform (GCP)**

  To access the dataset from Google Cloud Storage you need to define a class called Globals with several parameters related to a Google Cloud Platform (GCP) project and Google Cloud Storage (GCS) bucket. These parameters include the GCP project name, GCS bucket name, paths to CSV files containing metadata and labels, and the directory containing images within the bucket. The class also initializes some variables like a GCS storage client, bucket object, and DataFrames for the loaded CSV files. Finally, it prints out the values of some of the parameters defined in the class. Overall, this code is setting up global parameters and configurations for a data science or machine learning project that involves accessing data stored in a GCS bucket within a GCP project.

- **Create a DataFrame (DF)**

  After initializing a GCS client and bucket, loads 'metadata' and 'labels' CSV files from the bucket into pandas DataFrames, and merges them based on 'case_id', finally printing the length of the merged DataFrame.

3.5.4. Display a random Images

This step defines functions to display images associated with a case ID and optionally print condition labels. It uses the Pillow library to handle images and Matplotlib to display them. The display_image function loads an image from Google Cloud Storage (GCS) based on a provided image path and displays it using Matplotlib. The display_images_for_case function selects a random case from a df based on the provided case ID (if any) or selects a random case if none is provided. It then retrieves the image paths associated with that case and displays each image using the display_image function. Additionally, it optionally prints condition labels associated with the case. The code is designed to work with a df that contains information about cases, including their IDs, image paths, and condition labels. It seems to be part of a larger project related to image classification or analysis, where cases are associated with images and corresponding condition labels.

3.5.5. Identify Invalid Images

This step defines a function is_valid_image_path to check if an image path is valid by attempting to open the image using Pillow. It then applies this function to each image path column in the DataFrame Globals.cases_and_labels_df, adding a new column for each image path column to indicate whether the image path is valid or not. Finally, it prints the first few rows of the df with the validity columns. The missing image path is : "dataset/images/-2243186711511406658.png

3.5.6. Reverse Engineering

This step defines a python script that reverses one-hot encoding by mapping binary encoded columns back to their original categorical values, iterating over specified groups and creating new df columns for each group's categorical values. It updates these columns with category names where the corresponding one-hot encoded column is 'YES', effectively restoring the original categorical values.

3.5.7. Dropping one hot encoded columns

After applying reverse engineering on the one-hot encoded columns to obtain a normal categorical column, all the encoded columns are deleted because they don't provide meaningful information. The dropping of encoded is performed by initializing a list that is initially empty. Then, creating a for loop that iterates over all the columns in the df to check if the column name starts with any prefix in the one_hot_groups dictionary. If it does, it is added to the list. Finally, all the columns in this list are dropped.

3.5.8. Feature engineering process

In this phase we calculates the most common value for each row across multiple sets of columns related to dermatologist gradings for skin conditions and Fitzpatrick skin types in a DataFrame. It defines a function to calculate the mode for each row and applies it to each set of columns using the apply function along the rows (axis=1) of the DataFrame, generating new columns with the most common values for each set of columns.

3.5.9. Impute Missing Values & Fix Unbalanced Data

First, the categorical labels are decoded using LabelEncoder to convert them into numerical values .After this, the missing values are imputed in the encoded column with the most frequent value using SimpleImputer. Then, the imputed numerical values are decoded back to their original categorical labels using LabelEncoder. Finally, the DataFrame is updated with the imputed categorical values in a new column.

3.5.10. Dropping Unneeded Columns

Because they either contain mostly null values or are not relevant for your analysis. The indices correspond to the following columns: 'case_id', source, release, year: These seem to be identifiers or metadata that are not necessary for your analysis. 'race_ethnicity_two_or_more_after_mitigation': Contains mostly null values. 'dermatologist_gradable_for_skin_condition_': These columns have many null values and might not be useful for your analysis. 'dermatologist_skin_condition_on_label_name', 'dermatologist_skin_condition_confidence, weighted_skin_condition_label', 'dermatologist_fitzpatrick_skin_type_label_': Likely not relevant or redundant for your analysis. 'gradable_for_monk_skin_tone_',' monk_skin_tone_label_': These columns seem specific to certain analyses and are not relevant to your current task. By dropping these columns, you're likely simplifying your dataset to focus on the most relevant features for your analysis, which can improve model performance and reduce computational overhead.

3.5.11. Renaming columns and its values

In this step, the unclear column names are renamed to make them easier to recognize. Also values in some columns are renamed to clarify their meaning for the reader. For example, rename columns to more descriptive names, e.g., 'fitzpatrick_skin_type' to 'fitzpatrick_scale', and 'image_1_path' to 'image_name'. Split and clean the 'textures' column to remove the prefix and convert values to lowercase. Replace underscores in the 'textures' column with spaces for readability.

3.5.12. Data Visualization

In this step includes creating visuals to gain insights, such as detecting data bias, identifying missing entries, and determining if any changes are needed in the DataFrame.

3.5.13. Data Extraction

This step contains

- Creating a function to delete records from the DataFrame that don't have corresponding images in the image folder, based on the image_name column.
- Validate that each record has an image in the images folder.
- Preprocess the image data.
- Get 15 valid sample records.
- Create another DataFrame called Val_samples to store these samples for further research.

4. Experiments: specifications, results and discussion

4.1. Machine setup

4.2. Parameters setup

Table 2 outlines the parameter configurations for the compared algorithms. Following the findings of Arcuri and Briand (Arcuri & Fraser, 2013), which indicate that default parameter values suffice for fair algorithm comparisons, we adopt default values to ensure impartiality. Using default settings mitigates potential bias from algorithm-specific tuning, providing a more objective performance evaluation. To enhance the robustness and reliability of the comparison, each simulation is conducted independently 30 times. This repetition accounts for variations in performance due to random initialization or stochastic processes, leading to a more precise evaluation of the algorithms' efficiency and effectiveness.

Table 2
Parameter setting of compared algorithms

Algorithm	**Parameter setting**
Common Settings	Population size: $N = 30$ Maximum fitness evaluation: $max_{FE} = 30.000$ Dimensions $Dim = 10$ Number of independent runs : 30
WOA	α decreases linearly from 2 to 0 $a2$ linearly decreases from -1 to -2
GWO	a decreases linearly from 2 to 0
HHO	$\beta = 1.5$
HBA	$\beta = 3$ $W = 0.8$ $p = 0.03$
BWO	$B_F = (0, 0.5)$
RSA	
SBOA and mSBOA	default

4.3. Evaluation measures

4.4. First series experiments: Implementation of mSBOA for global optimization using CEC 2022 engineering benchmark functions

4.4.1. CEC 2022 Benchmark Test Functions

The CEC 2022 benchmark functions, among the latest and most challenging, are used to evaluate the effectiveness of the proposed mSBOA algorithm. This test suite includes 12 benchmark functions extensively utilized in optimization research. Recognized as a contemporary problem set for optimization algorithm assessment, CEC 2022 comprises various functions designed to address different optimization challenges. Table 3 provides a detailed overview of the characteristics of the CEC 2022 functions, while Figure 2 presents their two-dimensional visualizations. In addition to standard benchmark functions, the CEC 2022 suite features hybrid and composite functions. Hybrid functions split variables into subcomponents, applying different base functions to each, which mirrors real-world optimization problems where subcomponents exhibit distinct properties. Composite functions, on the other hand, create a more complex optimization landscape by combining shifted and rotated versions of various base functions. This method captures the intricacies of real-world problems and ensures continuity around both global and local optima. By utilizing the CEC 2022 test suite, which includes a diverse range of benchmark functions-standard, hybrid, and composite-the performance of the mSBOA algorithm can be comprehensively evaluated. This thorough assessment demonstrates the algorithm's effectiveness in addressing various optimization challenges.

Table 3
CEC 2022 test functions.

Function No.	**Description**	**Range**	F_{min}
F1	Shifted and full Rotated Zakharov Function	[-100, 100]	300
F2	Shifted and full Rotated Rosenbrock's Function	[-100, 100]	400
F3	Shifted and full Rotated Expanded Schaffer's f6 Function	[-100, 100]	600
F4	Shifted and full Rotated Non-Continuous Rastrigin's Function	[-100, 100]	800
F5	Shifted and full Rotated Levy Function	[-100, 100]	900
F6	Hybrid Function 1 (N = 3)	[-100, 100]	1800
F7	Hybrid Function 2 (N = 6)	[-100, 100]	2000
F8	Hybrid Function 3 (N = 5)	[-100, 100]	2200
F9	Composition Function 1 (N = 5)	[-100, 100]	2300
F10	Composition Function 2 (N = 4)	[-100, 100]	2400
F11	Composition Function 3 (N = 5)	[-100, 100]	2600
F12	Composition Function 4 (N = 6)	[-100, 100]	2700

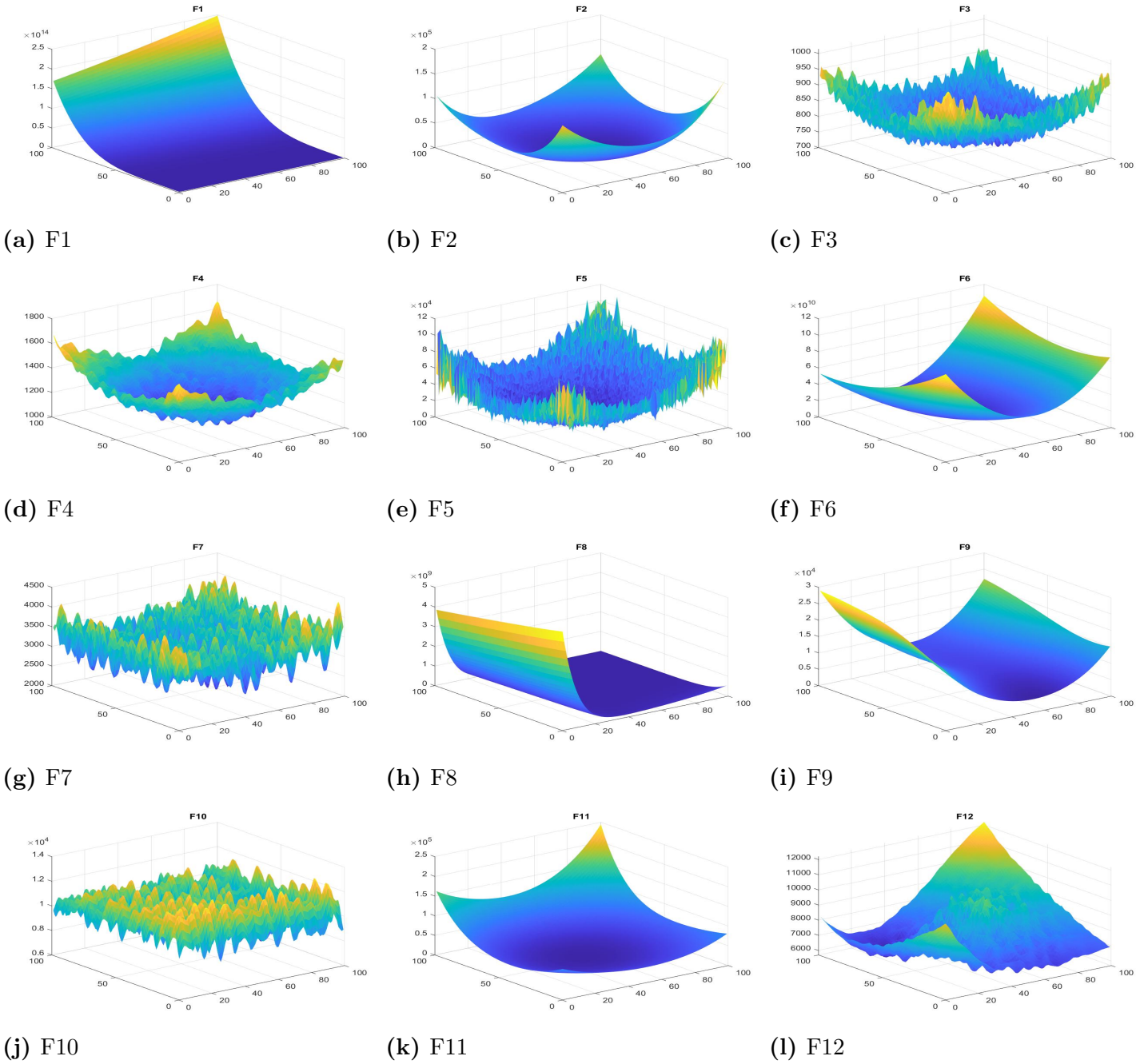
4.4.2. Performance measurements of mSBOA on CEC'2022 test functions

The effectiveness of the LFR-COA method is evaluated using a set of performance criteria. These measures have the following definitions:

- **Statistical mean:** The statistical mean is calculated as the average of the cost values obtained from multiple runs as shown in Eq. (7).

$$Mean = \frac{1}{R_n} \sum_{j=1}^{R_n} Fitt_b^i \tag{7}$$

Figure 2 The 2D of the CEC 2022 benchmarks.



(a) F1 **(b)** F2 **(c)** F3

(d) F4 **(e)** F5 **(f)** F6

(g) F7 **(h)** F8 **(i)** F9

(j) F10 **(k)** F11 **(l)** F12

- **Worst value:** The worst value represents the maximum fitness value achieved by the algorithm among all the runs. It is obtained using the Eq. (8):

$$WORST = \max_{1 \leq j \leq R_n} Fitt_b^i \tag{8}$$

- **Best value:** The algorithm's lowest fitness value, determined by calculating the average of all runs, is the best value. It is calculated using Eq. (9)

$$BEST = \min_{1 \leq j \leq R_n} Fitt_b^i \tag{9}$$

- **Standard deviation (STD):** The standard deviation measures the dispersion or variability of the cost values obtained from the different runs. It is computed using the Eq. (10):

$$STD = \sqrt{\frac{1}{R_n - 1} \sum_{j=1}^{R_n} (Fitt_b^i - Mean)^2} \tag{10}$$

where the number of runs overall is denoted by R_n.

4.4.3. Statistical results analysis
4.4.4. Statistical comparison using Boxplots
4.4.5. Convergence behavior analysis
4.4.6. Qualitative Metrics Discussion

Tracking the movements of agents during algorithm convergence and search optimization can offer valuable insights. Figure **??** presents the qualitative analysis of the proposed algorithm, illustrating agent behavior within a 2-D function space. The figure includes convergence plots, search histories, and mean fitness histories to provide a comprehensive view of the search process. We can better understand the proposed algorithm's efficiency and performance by analyzing these behaviors.

4.5. Second series experiments: Implementation of ESBOA for SCINE optimization using CEC 2022 engineering benchmark functions
4.6. Statistical assessments

5. Conclusion and future plans

Acknowledgements

References

Aarts, E., Aarts, E. H., & Lenstra, J. K. (2003). *Local search in combinatorial optimization*. Princeton University Press.

Arcuri, A., & Fraser, G. (2013). Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, *18*, 594–623.

Emam, M. M., Houssein, E. H., Tolba, M. A., Zaky, M. M., & Hamouda Ali, M. (2023). Application of modified artificial hummingbird algorithm in optimal power flow and generation capacity in power networks considering renewable energy sources. *Scientific Reports*, *13*, 21446.

Gao, W.-f., Liu, S.-y., & Huang, L.-l. (2013). A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Transactions on Cybernetics*, *43*, 1011–1024.

Houssein, E. H., Emam, M. M., & Ali, A. A. (2022). An optimized deep learning architecture for breast cancer diagnosis based on improved marine predators algorithm. *Neural computing and applications*, *34*, 18015–18033.

Tizhoosh, H. R. (2005). Opposition-based learning: a new scheme for machine intelligence. In *Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on* (pp. 695–701). IEEE volume 1.

Zhang, H., Heidari, A. A., Wang, M., Zhang, L., Chen, H., & Li, C. (2020). Orthogonal nelder-mead moth flame method for parameters identification of photovoltaic modules. *Energy Conversion and Management*, *211*, 112764.