A THESIS

ON

DEVELOPMENT OF HYBRID EVOLUTIONARY ALGORITHMS

BY

MS. TEJNA KHOSLA

(2K17/PHDCO/07)

UNDER THE SUPERVISION OF

PROF. O. P. VERMA

HEAD OF DEPARTMENT

DEPARTMENT OF ELECTRONICS & COMMUNICATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE

DOCTOR OF PHILOSOPHY
IN
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY, DELHI
INDIA

2024

DECLARATION

I hereby declare that the thesis entitled "Development of Hybrid Evolutionary Algorithms", submitted by Tejna Khosla for the award of the degree of *Doctor of Philosophy* to Delhi Technological University is a record of bonafide work carried out under the supervision of Dr. O. P. Verma, Professor, Department of Electronics and Communication, Delhi Technological University, Delhi.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Tejna Khosla
Roll No. 2k17/Ph.DCO/07
Department of Computer Science and Engineering
Delhi Technological University

Date:

CERTIFICATE

This is to certify that the thesis entitled "Development of Hybrid Evolutionary Algorithms" submitted by Ms. Tejna Khosla, Roll no. 2k17/Ph.DCO/07 as a part-time scholar in the Department of Computer Science and Engineering, Delhi Technological University, for the award of the *Doctor of Philosophy degree*, is a record of bonafide work carried out by her under my supervision. The content of this report has not been submitted and will not be submitted in part or in full for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and, in my opinion, meets the necessary standards for submission.

Place: Delhi, India
Date:

Dr. O.P. Verma
Professor
Delhi Technological University

ABSTRACT

Evolutionary Algorithms (EA), particularly swarm-based algorithms, have demonstrated their adaptability and efficacy in optimization across various domains. Although there are several algorithms inspired by nature, each with advantages and disadvantages, they are all highly effective. These algorithms provide faster convergence and global optimum solutions in complex problems, but they sometimes become stuck in local optima, which reduces their overall effectiveness.

This research work contributes to the optimization of real-world problems by modifying and hybridizing conventional swarm-based EA. To address various optimization issues, four algorithms are created.

Firstly, by combining features of the traditional Bacterial Foraging Algorithm (BFA) and the Firefly Algorithm (FA), a hybrid algorithm known as the Bacterial Foraging Algorithm-Firefly Algorithm (BFA-FA) is developed. The performance of the BFA-FA algorithm has been evaluated using traditional, noisy nonlinear, single-peak, multipeak, and contemporary CEC benchmark functions. The algorithm also combines adaptive and leadership strategies. Furthermore, the BFA-FA algorithm successfully addressed two structural design problems: the cantilever beam design problem and the three-bar truss design problem.

Secondly, a parameter-free algorithm called the particle swarm optimization-butterfly optimization Algorithm (PSOBOA) is developed to deal with constraints. Using a parameter-free penalty function, the PSOBOA algorithm successfully manages constraint violations, guaranteeing robustness throughout the search process. The PSOBOA algorithm also incorporates a self-adaptive strategy in both the particle swarm optimization algorithm (PSO) and the butterfly optimization algorithm (BOA) to enable a smooth transition from exploration to exploitation without requiring user intervention. A conditional approach is added to BOA's local and global searches to improve convergence rates and counteract local optima stagnation. The PSOBOA algorithm is successfully applied to structural optimization problems with various objectives, decision variables, and constraints, such as pressure vessel design and welded-beam design. A comprehensive comparison has been conducted with six conventional algorithms using performance indicators such as mean, standard deviation, rank, and time.

Thirdly, PSO and the Chameleon Search Algorithm (CSA) are combined to create a hybrid algorithm called the Chameleon Search Algorithm-Particle Swarm Optimization (CSAPSO). To improve optimization efficiency, the CSAPSO algorithm integrates Opposition-Based Learning with Hybrid. The proposed CSAPSO algorithm has been evaluated on twelve chest X-ray (CXR) and COVID-19 CXR images. CSAPSO's performance is assessed by contrasting its outcomes with other state-of-the-art optimization algorithms and other deep learning models using measures like Root Mean Square Error (RMSE), Peak signal-to-noise ratio (PSNR), and Structure Similarity Index (SSIM), Classification Accuracy, Area Under Curve (AUC).

Finally, a hybrid optimization algorithm called the Opposition-based Particle Swarm Algorithm-Grey Wolf Optimization algorithm (Opp-PSOGWO) is developed by integrating Figurate Opposition-Based Learning (OBL) into the hybridization of PSO and Grey Wolf Optimizer (GWO). Opp-PSOGWO generates solutions in opposite directions, in the Fibonacci sequence, leading to an optimal initial population with increased diversity. The resultant hybrid algorithm can escape local optima traps, and Figurate Opposition-based Learning speeds up the search for the best solutions. Nine traditional benchmark functions (unimodal and multimodal) were used to test the Opp-PSOGWO model. Then, it has been compared to its parent algorithms, PSO and GWO, taking into account performance metrics, including mean, standard deviation, and CPU time.

In conclusion, this research provides four unique algorithms that contribute to swarm-based EA techniques. Compared to traditional algorithms, the hybrid BFA-FA, parameter-free PSOBOA, hybrid CSAPSO, and Opp-PSOGWO algorithms enhance optimization performance in various real-world applications.

ACKNOWLEDGEMENT

With great pleasure and a deep sense of gratitude, I would like to express my sincere thanks to my supervisor **Prof. O.P. Verma**, HOD (Department of ECE), Delhi Technological University. Without his constant motivation and encouragement, this research would not have been successful. He has always been my pillar of strength.

I would like to take this opportunity to thank **Prof. Vinod Kumar**, the HOD (Department of CSE) and all faculty members of the Computer Science and Engineering Department, Delhi Technological University for their encouragement and support.

I would like to extend my profound sense of gratitude to **my father-in-law Sh. Arun Kumar Khosla, mother-in-law Smt. Monika Khosla, father Sh. Sham Patodi, mother Smt. Alka Patodi, brother Tarun Patodi, sister-in-law Sakshi Patodi** for all the sacrifices they made during my research and also for providing me with moral support and encouragement whenever required.

Last but not least, I would like to thank my husband **Sahil Khosla**, my daughter **Maithilee Khosla**, my son **Daivik Patodi**, and my friends **Dr. Anshu Khurana** and **Dr. Suruchi Mittar** for their constant encouragement and moral support along with patience and understanding.

Ms. Tejna Khosla

Department of Computer Science and Engineering

Delhi Technological University

Delhi-110042

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

EA	Evolutionary Algorithms
BFA	Bacterial Foraging Algorithm
FA	Firefly Algorithm
BFA-FA	Bacterial Foraging Algorithm-Firefly Algorithm
PSOBOA	Particle Swarm Optimization-Butterfly Optimization Algorithm
PSO	Particle Swarm Optimization Algorithm
BOA	Butterfly Optimization Algorithm
CSA	Chameleon Swarm Algorithm
CSAPSO	Chameleon Swarm Algorithm-Particle Swarm Optimization
CXR	Chest X-Ray
RMSE	Root Mean Square Error
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural similarity index measure
AUC	Area Under Curve
Opp-PSOGWO	Opposition based PSO and Grey Wolf Optimizer
OBL	Opposition based learning
GWO	Grey wolf optimizer
NIOA	Nature Inspired Optimization Algorithm
GA	Genetic Algorithm
ES	Evolutionary Strategies
EP	Evolutionary Programming
GP	Genetic Programming
DE	Differential Evolution
SI	Swarm Intelligence
CS	Cuckoo Search
BA	Bat Algorithm
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization

SA	Simulated Annealing
HS	Harmony Search
GSA	Gravitational Search Algorithm
BBBC	Big Bang-Big Crunch
PSODE	Particle Swarm Optimization with Differential Evolution
PSOGA	Particle Swarm Optimization with Genetic Algorithm
PSOGSA	Particle Swarm Optimization with Gravitational Search Algorithm
PSOGWO	Particle Swarm Optimization with Grey Wolf Optimizer
SSA	Salp Swarm Algorithm
COP	Constrained Optimization Problem
UDE	Unified Differential Evolution
GGA	Gradient-based Genetic Algorithm
EOGWO	elite-opposition based learning GWO
BBO	Biogeography-Based Optimization
HRCT	High-Resolution Computed Tomography
MR	Magnetic Resonance
SSO	Shallow Swarm Optimization
SVM	Support Vector Machine
E. coli	Escherichia Coli
MBO	Monarch Butterfly Optimization
BF-PSO	Hybrid BFA and PSO
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
NRMSE	Normalized Root Mean Square Error
SOS	Symbiotic Organisms Search
MMA	Method of Moving Asymptotes
GCA	Generalized Convex Approximation
DEDS	Differential evolution with dynamic stochastic selection
MBA	Mine blast algorithm
GOA	Grasshopper Optimization algorithm
ROI	Regions of interest

MCET	Minimum Cross-Entropy Thresholding
SCA	Sine Cosine Algorithm

PUBLICATIONS

International Journals:

1. Khosla, T., and Verma, O. P. (2022). An adaptive rejuvenation of bacterial foraging algorithm for global optimization. *Multimedia Tools and Applications*, 82. pages 1965–1993 (SCI/SCIE indexed).

2. Khosla, T., and Verma, O. P. (2023). Optimal threshold selection for segmentation of Chest X-Ray images using opposition-based swarm-inspired algorithm for diagnosis of pneumonia. *Multimedia Tools and Applications* (Available online)(SCI/SCIE indexed).

Under Review:

1. Tejna Khosla, O. P. Verma, "Particle Swarm Optimization improved by Salp Swarm Algorithm for optimal Training of Feed-forward Neural Networks", 2024, New Generation Computing.

Publication in International Conferences:

1. Khosla, T., and Verma, O. P. (2021, October). An Adaptive Hybrid Particle Swarm Optimizer for Constrained Optimization Problem. *In 2021 International Conference in Advances in Power, Signal, and Information Technology (APSIT)* (pp. 1-7). IEEE.

2. Tejna Khosla, O. P. Verma. An Efficient Particle Swarm Optimization with Fusion of Figurate Opposition-based learning and Grey Wolf Optimizer. *In 2022 International Conference on Computing, Communication, and Intelligent Systems* (pp. 209-213). IEEE.

CHAPTER 1

INTRODUCTION

1.1 Background

Evolutionary Algorithms (EA), also called Nature-Inspired Optimization Algorithms (NIOA), are a group of optimization algorithms that draw inspiration from natural phenomena. The optimization process involves selecting the optimal solution from a set of alternative ones. The objectives for optimization can vary, ranging from maximizing performance, sustainability, and efficiency, to minimizing cost, time, and energy consumption. EA employs heuristics along with learning strategy, memory, and solution history. They are also called Global Optimizers because of their ability to explore the solution space effectively without getting trapped in local optima.
Fig. 1.1 shows the transformation model of the nature-inspired optimization approach. It emphasizes the essential steps in modeling and using these algorithms for real-world applications. The process begins by observing any natural phenomenon that inspires the development of mathematical functions and equations. Finally, the model-based optimization approach is used to solve real-world applications.
EA provides a powerful way to handle challenging optimization problems by drawing on knowledge of nature. They offer an adaptable and flexible framework to address various issues in fields such as engineering, business, logistics, and healthcare. EA has been shown to have a lot of potential to improve system efficiency, resource use, and decision-making.

1.2 Characteristics of evolutionary algorithms

EA possesses several qualities and features that contribute to its strength and wide applicability. The following are the main characteristics of EA:

i Global optimizers: EA can identify the actual global optimum solution to optimization problems. EA investigates the whole solution space, in contrast to local optimization techniques, increasing the possibility of discovering the ideal solution.

ii Black-box approach: EA works on problems without needing explicit problem-

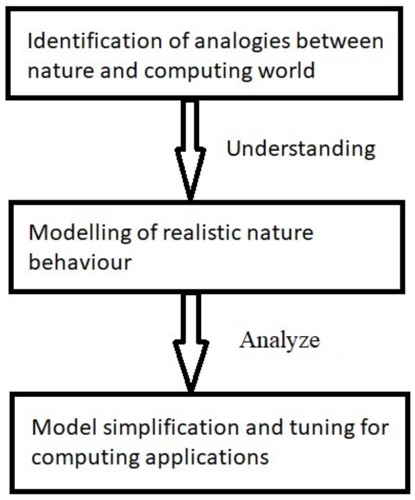

Identification of analogies between nature and computing world
Understanding
Modelling of realistic nature behaviour
Analyze
Model simplification and tuning for computing applications

Figure 1.1: Model of evolutionary algorithms

specific information or specialized knowledge. They approach problems as ”black-box”, which means that they do not require an in-depth understanding of the problem structure, making them adaptable and useful for a variety of problem domains.

iii Dealing with nonlinear and multimodal situations: EA is very good at dealing with non-linear and multimodal optimization issues. They are not constrained by the smoothness of the objective function or the presence of multiple optimal solutions because of their gradient-free nature. As a result, EA can discover global optima even in challenging environments and manage complex scenarios with discontinuities.

iv Stochastic components: Random walks and random numbers are examples of stochastic components included in the EA. EA can avoid the trap of local minima and explore a wider range of the solution space due to their random behavior. The stochastic nature of EA results in a variety of solutions for different runs or iterations from the same initial conditions.

v Exploration and exploitation: EA has a global and local search component that balances exploration and exploitation. The global search component allows for a thorough examination of potential solutions by examining new locations within the solution space. The local search element focuses on enhancing and taking advantage of potential areas close to previously found solutions. Using a variety of strategies, EA strikes a balance between exploration and exploitation, improving its ability to find the most effective solutions.

These features make evolutionary algorithms effective tools for solving optimization problems in a variety of fields. Their effectiveness and extensive applicability are a result of their ability to function globally, adapt to various problem structures, manage complexity, and strike a balance between exploration and exploitation.

1.3 General framework of evolutionary algorithms

With the study of various EA, a general framework can be developed that mainly all algorithms under the category follow. The step-by-step framework is as follows:

i Initialization: Begin with a population of solutions drawn at random. Since EA are mostly multi-agent systems, this step is essential since the initial population establishes the framework for the optimization procedure.

ii Fitness evaluation: Using a fitness function, determine the fitness of each solution in the population. The fitness function directs the search for ideal solutions by quantifying the fitness or objective value of each agent.

iii Generation update: Moving on to the next iteration or generation. In most circumstances, increase a counter that records the generation number.

iv Population evolution: Create a new population of solutions through multiple evolutionary processes at every generation. These processes encourage a balance between exploration and exploitation, including mutation, crossover, parent selection, randomization, and other techniques. By increasing variation and promoting the search for better solutions, these operations transform the population.

v Population replacement: In each generation, replace the current population with the newly generated population. Keep track of the best global solution found so far during the optimization process.

vi Termination criteria: Continue the procedure until a certain set of conditions are satisfied. These requirements can be to attain a predetermined number of generations, to reach a desirable degree of fitness, or to satisfy particular convergence conditions.

By following this step-by-step framework, EA can systematically explore and refine the population of solutions, gradually improving the performance and convergence of the optimization process. It offers a structured methodology that makes it feasible to efficiently explore the solution space while maintaining the goal of identifying the best solutions. It is crucial to remember that, while this framework captures the typical EA phases, different algorithms may have different implementation details and variants. Depending on the different traits and goals of various algorithms, the framework is modified and improved.

1.4 Hybrid evolutionary algorithms

A basic evolutionary algorithm may be sufficient to find the appropriate answer for a number of issues. However, it may not always be able to provide an efficient (optimal) solution for a given problem, as seen from the literature. It is evident from this that hybridizing evolutionary algorithms with other optimization algorithms, machine learning methods, heuristics, etc., is necessary. The following are some potential causes of hybridization:

i To enhance the evolutionary algorithm's performance (for instance, its speed of convergence).

ii Enhance the performance of the evolutionary algorithm.

iii Including the evolutionary algorithm in a more comprehensive system.

Some evolutionary algorithms focussed on for this work are described in the following section.

1.5 Preliminaries

This work mainly focuses on proposing hybrid algorithms using the following state-of-the-art algorithms.

i Bacterial foraging algorithm

ii Particle swarm optimization

iii Firefly algorithm

iv Salp swarm algorithm

v Grey wolf optimization

vi Butterfly optimization algorithm

vii Chameleon swarm algorithm

1.5.1 Bacterial foraging algorithm

BFA is an EA technique that works on the foraging nature of Escherichia coli abbreviated as E. Coli bacteria (1). It is designed to handle complex objective functions that are nondifferentiable and nongradient optimization problems. The four main mechanisms, viz, chemotaxis step, reproduction step, swarming, and elimination dispersal step are as follows:

i Chemotaxis

Two basic locomotive operations involved in chemotaxis are: Swim and tumble. The bacteria tumble when the flagella rotate clockwise. In this motion, there is the least displacement and, after completion, the bacteria are aligned along a random direction. On the other hand, the bacteria swim in a counterclockwise rotation. In this motion, the bacteria move forward in a particular direction. Let us consider the q^{th} chemotactic step for p^{th} bacteria of the r^{th} reproduction step of the elimination-dispersion kth of BFA. After each movement, the bacteria change their position according to Eq. (1.1) as given below.

$$pos(p,q+1,r,k) = pos(p,q,r,k) + C(p)\frac{\Delta(p)}{\sqrt{\Delta^T(p)\Delta(p)}} \tag{1.1}$$

where $\Delta(p)$ is the direction vector of the movement of the p^{th} bacteria in the current chemotactic step, and $C(p)$ is the size of the step in each chemotactic swim or tumble.

ii Reproduction

In this step, healthy bacteria remain, whereas the least healthy bacteria die. The global fitness value is updated in each reproduction step. Each bacterium that survives is split into two so that the total number of bacteria remains constant. The criteria for deciding the bacteria that will survive are decided by calculating the total health compared to the preceding reproduction, given by Eq. (1.2) below.

$$J_{health} = \sum F(p,q,r,k) \tag{1.2}$$

where F(p,q,r,k) is the fitness value of p^{th} bacteria for q^{th} chemotactic step, r^{th} reproduction step of the k^{th} elimination-dispersion step. The bacteria with minimum accumulated health function get selected for elimination.

iii Elimination and dispersion

This step occurs after a predefined number of reproduction steps with the intent of improving the global search. In this step, the bacteria are eliminated and dispersed in random positions to avoid getting trapped in the local optima. This happens according to the probability N_{eldis}, probability of elimination and dispersion.

iv Swarming

It has been noticed that several bacteria species including E.Coli form a stable spatio-temporal ring-shaped swarm in the presence of a semisolid nutrient medium. E. coli cells when stimulated by a high level of succinate release an attractive compound called aspartate that helps them aggregate in such groups. This cell-cell attraction can be mathematically represented as the Eq. (1.3) below.

$$\begin{aligned} J_{cc}(pos) &= \sum_{p=1}^{s} J_{cc}(pos(p,q,r,k)) \\ &= \sum_{p=1}^{s}(-d_{attractant} * exp(-w_{attractant}\sum_{d=1}^{D}(pos_d - pos(p,q,r,k)_d)^2)) \\ &+ \sum_{p=1}^{s}(h_{repellant} * exp(-w_{repellant}\sum_{d=1}^{D}(pos_d - pos(p,q,r,k)_d)^2)) \end{aligned} \tag{1.3}$$

where $J_{cc}(pos)$ is the fitness offset in the q^{th} chemotaxis or the cost function that is added to the original cost function; $d_{attractant}$, $w_{attractant}$, $h_{repellant}$, $w_{repellant}$ are the coefficients which determine the depth and width of attractant and height and width of repellant, which is to be selected properly; D is the number of dimensions; pos_d means the population coordinates in d-dimension; $pos(p,q,r,k)_d$ means the p^{th} bacterium coordinates in d-dimension.

1.5.2 Particle swarm optimization

The PSO algorithm was created by Kennedy and Eberhert and was motivated by the swarming behavior of a flock of birds (2). Usually, a set of potential solutions evolves until it finds the optimal one. The population of particles in the search space is randomly initialized at the beginning of this method. The position and velocity formulas are used to move the particles in the search space. During each iteration, each particle is influenced by its local best solution and directed toward the global best solution.
The particles try to modify their position and velocity as per Eq. (1.4) and Eq. (1.5).

$$v_i^{t+1} = wv_i^t + c_1 \times r_1 \times (Pbest_i - pos_i^t) + c_2 \times r_2 \times (Gbest - pos_i^t) \tag{1.4}$$

$$pos_i^{t+1} = pos_i^t + v_i^{t+1} \tag{1.5}$$

where w is the weighting function (inertia), v_i^t is the velocity of the ith particle at iteration t, $c_1 and\ c_2$ are the acceleration coefficients, $r_1\ and\ r_2$ are the random coefficients between [0,1], pos_i^t is the position of particle at iteration t, $Pbest_i$ is the best previous position concerning ith particle, and $Gbest$ is the best global solution so far for all particles.
The first element of Eq. (1.4) i.e., wv_i^t offers the opportunity of exploration while simultaneously maintaining the current direction, the second part i.e., $c_1 \times r_1 \times (Pbest_i - pos_i^t)$

provides the cognitive component (private thinking), and the last part i.e., $c_2 \times r_2 \times (Gbest - pos_i^t)$ offers the social component (collaborative approach). The sequence continues until the stopping criterion. Algorithm 1 provides a summary of the PSO steps.

Algorithm 1 Particle Swarm Optimization Algorithm

```
1. Set the control parameters
2. Do
3. For each particle
4. Calculate the fitness of the particle
5. As required, update $Pbest$
6. Update $Gbest$ as necessary
7. End for
8. Adjust the value of $w$
9. For each particle
10. Use Eq. (1.4) to update the velocity
11. Use Eq. (1.5) to update the position
12. End for
13. While the end condition is not satisfied
14. Return $Gbest$ as the global optimum value
```

1.5.3 Firefly algorithm

FA (3) is a successful nature-inspired optimization algorithm. In this algorithm, there are three main rules; 1) Fireflies get attracted to each other, independent of their sex, 2) Attractiveness is proportional to Brightness and vice versa, and for any two Fireflies, the movement takes place from less bright firefly to brighter firefly. If no firefly is brighter than a specific Firefly, it will randomly travel in any direction, 3) The brightness of a Firefly is evaluated by the objective function. The distance between any two Fireflies i and j whose positions are x_i and x_j, respectively, for dimensions D is given by the

Cartesian distance as in Eq. (1.6).

$$dist_{ij} = ||x_i - x_j|| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{1.6}$$

The Firefly's attractiveness is proportional to the intensity of light seen by the adjacent firefly. The attractiveness is determined as Eq. (1.7).

$$B(r) = \beta_0 e^{-\gamma dist^2} \tag{1.7}$$

where β_0 means the attractiveness at $dist = 0$ and γ is the light absorption coefficient at the source. The movement of a firefly i to another more attractive Firefly j is determined by Eq. (1.8):

$$x_i^{new} = x_i^{current} + \beta_0 e^{-\gamma dist_{ij}^2}(x_i^{current} - x_j) + \alpha(rand - \frac{1}{2}) \tag{1.8}$$

where the second term is due to attraction and the third term is due to randomization, $\alpha = [0, 1]$ being the randomization parameter, rand is a random generator of uniform distribution between 0 and 1, x_i^{new} is the new position of firefly x_i, and $x_i^{current}$ is the current position of firefly x_i. In this study, we set the light absorption coefficient $\gamma = 1$ and $\beta_0 = 1$.

1.5.4 Salp swarm algorithm

A novel Salp Swarm Algorithm (SSA) (4) takes inspiration from the foraging behavior of salp in chains using harmonized movements. This movement was mathematically modeled and tested in various optimization functions. SSA is a population-based approach that divides the population into two groups, Leader and followers. The leader salps are the pioneers in the chain and guide the remaining salps, who become the followers. In this way, the salps move together towards the global best position. A swarm X consisting of n salps in a space of dimensions d can be represented in the matrix as given in Eq. (1.9).

$$X_i = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \vdots & \vdots & \dots & \vdots \\ x_1^n & x_2^n & \dots & x_d^n \end{bmatrix} \tag{1.9}$$

The leader salp updates its position while moving towards the food source according to Equation Eq. (1.10).

$$x_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j) \times c_2 + lb_j) & c_3 \leq 0 \\ F_j - c_1((ub_j - lb_j) \times c_2 + lb_j) & c_3 > 0 \end{cases} \tag{1.10}$$

where x_j^1 denotes the position of the leader in j^{th} dimension. F_j is the food source that influences the movement of the salp lb_j refers to the lower bound, and ub_j refers to the upper bound, $c_2,\ c_3$are the random variables between the range [0,1]. c_1 is the controlling parameter to balance the exploration and exploitation phase and is calculated in Eq. (1.11).

$$c_1 = 2e^{-(\frac{4t}{max\_Iter})^2} \tag{1.11}$$

where t is the current iteration and max_Iter is the maximum number of iterations. After updating the position of the leader, the followers update their positions according to Eq. (1.12).

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}) \tag{1.12}$$

where $i \geq 2$ and x_j^i is the position of i^{th}follower along j dimension. The steps of the SSA algorithm are summarized in Algorithm 2.

Algorithm 2 Salp Swarm Algorithm

```
1. Initialize the controlling parameters.
2. While $t = 1 : max\_Iter$
3. Evaluate the fitness of the salp
4. Set $F$ as the best salp
5. Update $c_1$ using Eq. (1.11)
6. For each salp $x_i$
7. if (i==1)
8. The position of the leader salp is updated using Eq. (1.10)
9. else
10. The position of the follower salp is updated using Eq. (1.12)
11. endif
12. End for
```

```
    13. End while
    14. The best solution $F$ is returned
```

1.5.5 Grey wolf optimization

GWO is another metaheuristic developed by Mirjalili (5). It is inspired by the social hierarchy and the hunting mechanism of the grey wolves. Among the grey wolves, the leader is taken as alpha (α), the second level of wolves is called beta (β) which supports the leader wolves, the third level is called delta (δ), and the last level of wolves that have the lowest rank is called omega (ω).
Grey wolves hunt in three stages: (i) tracking, chasing, and approaching the prey; (ii) pursuing, encircling, and harassing the prey until it stops moving; and (iii) attacking the prey. X_p (the prey's position), X (grey wolf position at instantaneous iteration t), the vector coefficients A and C are calculated from (5).
The prey is located by alpha wolves. The beta and delta wolves support the alpha wolf in this process. The best solution is represented by alpha wolves, while the second and third best solutions are, respectively, shown by beta and delta wolves. The alpha, beta, and delta positions of other wolves are determined. This is formulated in Eq. (1.13).

$$\begin{aligned} D_\alpha &= |C_1 \times X_\alpha - X(t)| \\ D_\beta &= |C_1 \times X_\beta - X(t)| \\ D_\delta &= |C_1 \times X_\delta - X(t)| \end{aligned} \tag{1.13}$$

where X_α, X_β, and X_δ are the best three positions during each iteration, respectively. they are defined in Eq. (1.14) below.

$$\begin{aligned} X_1 &= |X_\alpha - a_1 D_\alpha| \\ X_2 &= |X_\beta - a_2 D_\beta| \\ X_3 &= |X_\delta - a_3 D_\delta| \end{aligned} \tag{1.14}$$

The new position of prey is calculated in terms of X_1, X_2 and X_3 as Eq. (1.15).

$$X_p\,(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{1.15}$$

where the mean of the positions of the top three wolves is used to compute the new location of the prey, indicated by $X_p\,(t+1)$.

To attach the prey, the value of $|A|$ should be less than 1. If it is more than 1, the wolves tend to diverge, showing exploration (6). However, when it is less than 1, it shows the exploitation ability of wolves leading to attack. Generally, the value of A lies in the range [-2a,2a], as the value decreases from 2 to 0.
The steps of GWO are summarized in Algorithm 3.

Algorithm 3 Grey Wolf Optimization Algorithm

```
1. Initialize the controlling parameters ($\alpha$, A and C)
2. Do
3. For each wolf
4. Calculate the fitness of all wolves
5. Calculate the values of $X_\alpha$, $X_\beta$ and $X_\delta$
6. Calculate $X_1$, $X_2$ and $X_3$ using Eq. (1.14).
7. Update the position of grey wolves
8. Update $\alpha$, A and C values
9. End for
10. Calculate the new position of the prey using Eq. (1.15).
11. While the end condition is not achieved
12. Return the best global optima.
```

1.5.6 Butterfly optimization algorithm

BOA is a recent metaheuristic developed by Satvir and Arora (7). The source of inspiration for this method is the foraging behavior of butterflies in search of food. The butterflies are the search agents used during the optimization process. Butterflies have sense receptors through which they sense the fragrance of food and move in that direction. This fragrance defines the fitness function of the BOA. Furthermore, the fragrance generated by a butterfly is carried to other butterflies around in the same search space. When the butterfly senses the fragrance from the best butterfly and starts moving toward it, this is known as the global search phase of the BOA. Also, when the butterfly is unable to sense the fragrance of any other butterfly and starts moving randomly, it is known as the local search phase of BOA.

The fragrance is updated with each movement of the butterfly. It depends mainly on three parameters, that is, the power exponent (a), stimulus intensity (I), and sensory modality (c). Sensory means measuring the energy form, and the modality is the raw input to the sensors. Modalities can be of different types such as light, smell, sound, temperature. However, in BOA, the fragrance modality is used. Intensity is the magnitude of the actual stimulus. In BOA, it is the value of the fitness function. The power exponent is the value at which the intensity is raised. It is responsible for regular expression, linear response, and response compression.
The fragrance is formulated as a function of the intensity of the stimulus in BOA, as shown in Eq. (1.16)

$$fr_i = cI^a \tag{1.16}$$

where fr_i is the perceived magnitude of fragrance by i-th butterfly, c is the sensory modality, I is the stimulus intensity, and a is the power exponent which tells the degree of absorption.
During the global phase, the butterfly moves towards the fittest butterfly according to Eq. (1.17).

$$posn_i^{t+1} = posn_i^t + \left(r^2 \times gb - posn_i^t\right) \times fr_i \tag{1.17}$$

Where $posn_i^t$ is the solution vector $posn_i$ for the i-th butterfly during iteration t, gb is the best solution so far in the current iteration, fr_i is the fragrance of the i-th butterfly, r is the random number in the range [0,1].
During the local phase, the butterfly moves randomly according to Eq. (1.18).

$$posn_i^{t+1} = posn_i^t + \left(r^2 \times posn_j^t - posn_k^t\right) \times fr_i \tag{1.18}$$

where $posn_j^t$and $posn_k^t$ are the solution vectors for the j-th and k-th butterfly. If they belong to the same swarm, the Eq. (1.18) becomes a random local walk. A switch probability p is used to switch between the global and local search phases. The steps of the BOA algorithm are summarized in Algorithm 4.

Algorithm 4 Butterfly Optimization Algorithm

```
1. Initialize the population of butterflies and controlling parameters
2. Evaluate the fitness of the butterflies and find the best solution
3. Define the switch probability $p$
4. Do
5. For each butterfly
6. Calculate the fragrance of the butterfly using Eq. (1.16)
```

```
7. Generate a random number $rand$ in the range [0,1]
8. If $rand < p$
9. Perform global search using Eq. (1.17)
10. Else
11. Perform local search using Eq. (1.18)
12. Endif
13. Evaluate new solutions
14. Update better solutions
15. End for
16. While the end condition is not fulfilled
17. Return the best global optima
```

1.5.7 Chameleon swarm algorithm

CSA is the nature-inspired algorithm developed by Braik et al. (8). It is inspired by the hunting behavior of chameleons in forests to sustain themselves. The chameleons have the capability of rotating their eyes 360 degrees and work independently of each other. The hunting mechanism of the chameleon consists of three distinct steps, namely; (i) tracking the prey, (ii) pursuing the prey with their eyes, and (iii) attacking the prey with the use of the tongue. Chameleons can roam the entire search space in search of prey and use their globular eyes to get a panoramic view of their surroundings. They use their long sticky tongue to catch the prey by the phenomenon of wet adhesion and entanglement. Their suction cup-like tongue traps the prey.
Initialization of chameleons (number and dimension) in the search space is done using Eq. (1.19).

$$A^i = l_j + r * (u_j - l_j) \tag{1.19}$$

where A^i represents the initial vector of the i^{th} chameleon, l_j, and u_j are the lower and upper bounds of the space along the j^{th} dimension, respectively, r is a uniformly generated random number in the range $[0,1]$. The position update strategy can be mathematically modeled using Eq. (1.20).

$$A^{i,j}_{t+1} = \begin{cases} A^{i,j}_t + p_1(BP^{i,j}_t - GP^j_t)r_2 + p_2(GP^j_t - A^{i,j}_t)r_1 & r_i \geq Pp \\ \\ A^{i,j}_t + \mu((u_j - l_j)r_3 + l^d_j)sgn(rand - 0.5) & r_i < Pp \end{cases} \tag{1.20}$$

where $A^{i,j}_{t+1}$ is the new position of i^{th} chameleon in dimension j^{th} for the iteration number $t+1$, $A^{i,j}_t$ is the position in the t^{th} iteration, $BP^{i,j}_t$ is the best position of i^{th}

chameleon so far, GP_t^j is the best global position in dimension j^{th} by any chameleon, p_1,p_2 are two positive numbers to control the exploration; r_1, r_2, r_3 are the random uniform numbers in the range [0,1], Pp is the probability of perceiving the prey. The balance between exploration and exploitation is achieved by tuning the adaptive parameters of μ according to Eq. (1.21).

$$\mu = \gamma^{(\frac{-\alpha t}{T})^{\beta}} \tag{1.21}$$

where $\gamma\alpha\beta$ are the exploration-exploitation control parameters, t is the current iteration, T is the maximum iteration.
Eq. (**??**) gives the new position update after detecting the prey by eye rotation.

$$A_{t+1}^i = Ar_t^i + \overline{A}_t^i \tag{1.22}$$

where A_{t+1}^i is the new position of the chameleon, Ar_t^i is the center of the current position before rotation and $\overline{A}_t^i$ is the rotating centered coordinates.
The velocity of the chameleon's tongue when it attacks the prey is according to Eq. (1.23).

$$v_{t+1}^{i,j} = \omega v_t^{i,j} + c_1(GP_t^j - A_t^{i,j})r_1 + c_2(BP_t^{i,j} - A_t^{i,j})r_2 \tag{1.23}$$

where ω is the inertia weight and defined in Eq. (1.24)

$$\omega = (1 - \frac{t}{T}^{(\rho\sqrt{\frac{t}{T}})}) \tag{1.24}$$

where ρ is a positive no. to manage exploitation, $v_t^{i,j}$ is the current velocity of the chameleon along j^{th} dimension.
Eq. (5.7) shows the position of the chameleon's tongue when it attacks the prey.

$$A_{t+1}^{i,j} = A_t^{i,j} + \frac{(v_t^{i,j})^2 - (v_{t-1}^{i,j})^2}{2a} \tag{1.25}$$

where $v_{t-1}^{i,j}$ being the previous velocity, a is the acceleration rate of the tongue's projection and is represented as Eq. (5.8).

$$a = 2590 * (1 - e^{-log(t)}) \tag{1.26}$$

Algorithm 5 summarizes the steps of CSA.

Algorithm 5 Chameleon Swarm Algorithm

```
1. Parameter definition: $Pp$: position update probability; $r_1, r_2, r_3, r_i$: random
   numbers between [0,1]; $u$: upper bound; $l$: lower bound; $d$: dimension; $\overline{A}_t^i$
   centre of the current position of chameleon $i$ at iteration $t$; $Ar_t^i$ : rotating
   centred coordinates of chameleon $i$ at iteration $t$; $T$: max iteration.
2. Initialize the controlling parameters.
3. Initialize the swarm of $n$ chameleons, $\overline{A}_t^i$i , and $Ar_t^i$ , velocity of chameleon's
   dropping tongue.
4. Calculate the position of chameleons.
5. While $t < T$
6. Define $\mu, \omega$, and $a$
7. For $i = 1$ to $n$
8. For $j = 1$ to $d$
9. If $r^i \geq Pp$ then
10. Update the position using equation
11. $A_{t+1}^{i,j} = A_t^{i,j} + p_1(BP_t^{i,j} - GP_t^j)r_2 + p_2(GP_t^j - A_t^{i,j})r_1$
12. Else
13. Update the position using equation
14. $A_t^{i,j} + \mu((u_j - l_j)r_3 + l_j^d)sgn(rand - 0.5)$
15. End if
16. End for
17. End for
18. For $i = 1$ to $n$
19. $A_{t+1}^i = Ar_t^i + \overline{A}_t^i$
20. End for
21. For i=1 to n
22. For j=1 to d
23. Update velocity using Equation (1.23)
24. Update position using Equation (5.7)
25. End for
26. End for
```

```
27. Use $u$ and $l$ to limit the position of chameleons
28. Evaluate and update the new position of chameleons
29. $t = t + 1$
30. end while
```

1.6 Classification of evolutionary algorithms

Based on inspiration from nature, EA is classified into three classes, namely, a) Evolution-based Algorithms, b) Bioinspired algorithms, c) other algorithms (9).

1.6.1 Evolution-based algorithms

Evolution-based Algorithms (10) take inspiration from the Darwinian theory of evolution. The main components of this algorithm are Representation (definition of individuals), Evaluation function (or fitness function), Population, Parent selection mechanism, Variation operators, recombination, and mutation, and Survivor selection mechanism (replacement). It is further categorized into 1) Genetic Algorithm (GA), 2) Evolutionary Strategies (ES), 3) Evolutionary Programming (EP), and 4) Genetic Programming (GP). John Holland developed the GA in 1975 (11), and it uses a binary representation of people. However, differential evolution (DE) and ES represent individuals using real values. Like GA, ES, and DE also employ mutation and crossover operators. Instead of using real numbers to represent an individual, GP uses the program (12). However, individuals are finite-state machines in EP, a system with several states and state transitions (13).

1.6.2 Bio-inspired algorithms

These algorithms take inspiration from biological systems and therefore constitute the major percentage of nature-inspired algorithms. They include two subcategories: 1) Swarm Intelligence algorithms, and 2) Nature-based algorithms.
Swarm intelligence (SI) (14) was developed by using the aggregate, promising behavior of many interacting agents that follow some simple principles. The collective behavior of social insects like ants, fireflies, and bees as well as other animal societies like flocks of birds or fish like the cuckoo, bat, etc. served as the basis for all SI-based algorithms, which are multi-agent/population-based. Many algorithms have been developed using SI systems, for instance, PSO was inspired by the swarming behavior of fish and birds, while the development of the FA was based on the flashing behavior of swarming fireflies, the Cuckoo Search (CS) was based on the brooding parasitism of some cuckoo species, and the Bat Algorithm (BA) utilized the echolocation of bat foraging. While the Artificial Bee Colony (ABC) algorithm and other bee algorithms are all based on

the foraging behavior of honey bees, Ant Colony Optimization (ACO) utilizes the interaction of social insects (such as ants). Among the most popular and successful EA are SI-based algorithms. This class of algorithms' success may be attributed to its tendency to distribute information among the population's individuals, which enables self-organization, coevolution, and learning to take place while searching and contribute to more accurate and ideal outcomes. Another reason is to do parallelism, and as a result, large-scale optimization is more feasible from an implementation standpoint.

1.6.3 Other algorithms

Recently, certain optimization algorithms that are neither bioinspired nor based on evolution have been developed employing various qualities, such as emotional and social ones. As a result, these algorithms fall under a different class. Differential search algorithm, social-emotional optimization, league championship algorithm, etc. are a few examples of these algorithms (15).

1.7 Principles, advantages, and limitations of conventional EA

1.7.1 Principles

The principles of EA can be summarized as follows:

i Nature as Inspiration: Biological, ecological, physical, or social systems in nature are the sources of inspiration for EA. They seek to mimic the collective, adaptive behavior, and survival techniques seen in diverse natural systems.

ii Population-Based Search: Instead of focusing on a single agent in the solution space, EA frequently uses a population of potential solutions. This population-based strategy allows for parallel and dispersed exploration, allowing the algorithm to effectively cover a larger portion of the solution space.

iii Stochastic and Probabilistic Elements: To introduce randomness in the search process, EA typically uses stochastic elements, such as randomization and probabilistic transitions. The algorithms can avoid local optima and more thoroughly explore the solution space because of these factors, which encourage exploration.

iv Balance Between Exploration and Exploitation: EA work to achieve a balance between exploration, finding new regions in the solution space, and exploitation, improving the already promising regions. This balance ensures an in-depth assessment of potential solutions while quickly advancing to the most optimal one.

v Self-Organizing and Adaptive: EA frequently includes adaptive features that allow the optimization algorithms to change their search techniques as needed. The algorithms can effectively optimize under changing situations because of self-organizing behaviors.

vi Global Search Capabilities: EAs are intended to be global optimizers, searching the solution space for the real global optimal solution as opposed to becoming stuck in local optima. They are suitable for resolving challenging multimodal optimization issues because of this capability.

vii Scalability: EA can effectively handle complex, high-dimensional optimization challenges. Due to their distributed and parallel architecture, they can successfully address complex optimization problems.

1.7.2 Advantages

Evolutionary Algorithms have a number of distinctive characteristics that make them both highly effective and desirable for solving challenging optimization problems. One of their main advantages is that they can perform global optimization, which enables them to look for the optimal solutions throughout the whole solution space without being stuck in local optima. Due to its stochastic nature and population-based search, EA demonstrate versatility and robustness, enabling them to tackle a variety of problem types, restrictions, and solution spaces. Furthermore, they work well for issues involving nondifferentiable or discontinuous objective functions due to their derivative-free nature. Additionally, EA exhibits scalability and parallelism, enabling them to successfully handle high-dimensional and massive optimization jobs. In EA, the balance between exploration and exploitation enables them to efficiently converge towards the best answer while exploring a wide range of potential solutions. Additionally, their imitation of heuristics derived from nature enables creative and effective optimization techniques. Due to these multiple benefits, EAs have grown significantly in popularity and are frequently utilized in a variety of industries to effectively and efficiently address problems in the real world.

1.7.3 Challenges

There are many problems in real-world problems where efficient searches have to be performed in complex spaces to attain an optimal solution.

i Problem definition: Despite the effectiveness and popularity of EA, there are still numerous challenges with these algorithms, especially from a theoretical point of view. Although researchers are aware of the fundamental principles underlying how these algorithms operate in reality, it is unclear why and exactly how they operate.

ii Building the right mathematical model: All evolutionary algorithms must be mathematically analyzed within a unified model to fully understand their convergence, rate of convergence, stability, and robustness. This framework requires the fusion of many mathematical, stochastic, and numerical methodologies to enable us to examine algorithms from a variety of angles.

iii Parameter Adaptation: The number of parameters in different EA vary greatly depending on the algorithm. These parameters affect the performance of the algorithm. If the parameter adaptation is done using brute-force technique, it takes longer. Self-adaptation is always a good idea to improve performance. However, specifying the rules for parameter adaption is always challenging.

iv Benchmarking: Benchmark functions are a set of test functions that are used to test the performance of any novel EA compared to other state-of-the-art algorithms. They are used mainly to study the efficacy of the algorithm with respect to its convergence behavior, exploration ability, exploitation ability, stability, and robustness. However, to decide which benchmark function to use, is difficult. The main point is that the benchmark functions are generally smooth and defined on regular domains, and they are unconstrained or with simple constraints. Therefore, by validating the algorithm using the benchmark function, there is no guarantee that the algorithm will perform equally well on real-world problems as well. Some algorithms are better at benchmark test functions but do not give good results in real-world applications, and vice versa.

v Performance Measures: Deciding the performance measures that are appropriate for the algorithm is equally important. The performance results give the conclusive statement about the behavior of any algorithm. Mostly the EA are concerned with achieving accurate global optimum, computational time, convergence, success rates, mean value, standard deviation, and so on. Another consideration here is that, due to the stochastic nature, the algorithm does not give the same results every time, so the performance metrics are observed for a large number of runs, keeping the population size constant for all the comparative algorithms.

vi Scalability of the Algorithm: The algorithm is considered effective and efficient if it can be used to solve a variety of real-world problems. As per the no-free-lunch theorem, it is known that an algorithm cannot deal with all the real-world problems in the same way. Another consideration is that if the scale of the problem changes from low-dimensional to higher dimensions, the algorithm loses its efficiency. So, it is a bigger challenge to deal with a wider range of applications and with different dimensions.

vii Hybridization: Hybridizing has always been the best practice for solving optimization problems where the features of two algorithms are incorporated and the resultant hybrid algorithm overcomes the weaknesses of both parent algorithms. However, choice of conventional algorithms for the hybrid is an important task and should be in such a way that the time complexity does not increase exponentially.

1.8 Motivation

The research presented in this thesis focuses on the development and evaluation of four hybrid evolutionary algorithms: BFAFA, PSOBOA, CSAPSO, and Opp-PSOGWO.
The motivation behind this research comes from the existing challenges faced by conventional optimization algorithms. Although conventional methods have made valuable contributions, they often have trouble dealing with complex issues that include constraints as well as nonlinear and multimodal objective functions. Additionally, these algorithms can suffer from premature convergence or get stuck in local optima, which limits their optimization effectiveness.
The algorithms proposed in this thesis are modifications and hybridization of traditional evolutionary techniques to overcome the above-mentioned limitations. These algorithms improve optimization performance, improve convergence rates, improve search space exploration, and address the difficulties of real-world problems by combining the advantages of different evolutionary algorithms and introducing unique strategies. By creating innovative hybrids and adaptations, the main goal of research is to advance in the field of optimization algorithms. These algorithms are made to be more flexible, to reach convergence more quickly, and to be more effective in handling a variety of real-world optimization problems. The goal of this research is to advance the state-of-the-art in optimization and give practitioners strong tools for handling challenging problems in a variety of fields by examining the possibility of hybridization and innovative methods.
This thesis aims to solve the drawbacks of standard optimization techniques by proposing and evaluating four hybrid evolutionary algorithms. They are driven by the need to improve the performance of optimization, overcome difficulties, and provide efficient solutions to problems encountered in the real world. This research advances the field of evolutionary algorithms through the development and analysis of these hybrid algorithms and offers useful knowledge for future optimization studies.

1.9 Problem statement

Even though evolutionary algorithms have shown promising results in addressing the optimization problems, more work is needed to develop and refine them so that they perform better and are more widely applicable. Many evolutionary algorithms find it

difficult to adequately explore the search space, which limits their performance to find globally optimal solutions. The search capabilities of the algorithms need to be improved by providing the approaches that encourage deeper exploration.
In optimization algorithms, slow convergence is a frequent problem that lengthens the optimization process and increases the computation time. The goal of this research is to improve and adjust the algorithms' parameter settings to accelerate convergence and solve the performance-related issues.
Multiple constraints and large-scale dimensions are common components of real-world optimization issues. It may be too complex for current algorithms to handle it correctly. The goal of this study is to investigate and validate how well the suggested algorithms perform when applied to complex and constrained optimization issues.
Benchmarks are required to assess the effectiveness and adaptability of optimization algorithms. It is necessary to create hybrid evolutionary algorithms with robust optimization performance that can be adjusted to a variety of generic test scenarios. The goal of the research is to hybridize and modify evolutionary algorithms that show better performance and suitability in handling common test scenarios.
This research aims to expand the field of evolutionary algorithms, overcome existing constraints, and provide practitioners with better tools to handle complicated optimization problems more effectively and efficiently by addressing these issues through the models presented.

1.10 Research objectives

The research objectives of this thesis are as follows:

- i To enhance search capabilities of NIOA by exploring search space effectively.
- ii Modifying/Enhancing the existing state-of-art by Parameter Tuning.
- iii Exploring and validating the performance over Large scale and Constrained Optimization.
- iv Hybridizing/Developing the evolutionary algorithms and substantiating their suitability in general test problems.

The objectives are achieved through the effective implementation of the following:

- i The algorithm's search capabilities are improved by enhancing the search space exploration.
  The algorithms are developed to incorporate innovative strategies and hybridization to improve the algorithms' search exploration, leading to a more comprehensive exploration of potential solutions. The development of the BFAFA, PSOBOA,

CSAPSO, and Opp-PSOGWO algorithms, which seek to enhance the search exploration of the algorithms through various hybridization and modifications, have addressed this issue.

ii The convergence speed of algorithms is optimized through parameter tuning.
The research aimed to modify and optimize the parameter settings to achieve faster convergence. The convergence speed was measured based on the average mean of individual fitness over successive generations. The BFAFA and PSOBOA algorithms, which focus on parameter adjustment to achieve quick convergence, explicitly targeted this goal.

iii The performance of algorithms is validated on large-scale and constrained optimization problems.
This objective was aimed at assessing the suitability, robustness, and effectiveness of these hybrid algorithms in solving real-world optimization problems with complex characteristics. The BFAFA and PSOBOA algorithms, which are evaluated on difficult scenarios to gauge their applicability, robustness, and efficiency in solving real-world optimization problems, addressed this issue.

iv The EA is improved and hybridized to enhance their suitability for general test problems.
The improved suitability and performance of the novel hybrid approach to solving a wide range of general test problems, highlighting its adaptability and robustness, was demonstrated using Opp-PSOGWO and CSAPSO.

1.11 Thesis organization

This thesis is organized into seven chapters, each focused on specific aspects of the research and contributing to the understanding and evaluation of the proposed nature-inspired optimization models. The organization of the thesis is as follows:

- **Chapter 1: Introduction**
  Chapter 1 reports the general description of the research topic, motivation, and objectives. It reviews the principles, advantages, and limitations of conventional optimization algorithms. It outlines the problem statement and research objective and introduces the proposed hybrid evolutionary algorithms. Finally, the layout of the thesis is also discussed.

- **Chapter 2: Literature review**
  Chapter 2 reviews the existing literature related to the topic of research. The study discusses the background and significance of EA and provides an overview

of relevant studies and research work in the field. It also identifies the research gap and justifies the need for the proposed hybrid algorithms and modifications.

- **Chapter 3: Hybrid bacterial foraging algorithm-firefly algorithm (BFAFA)**
  Chapter 3 presents a hybrid EA with parameter adaptation. Two algorithms named BFA and FA are used along with the leadership and adaptation strategy. After extensive testing and comparison with methodologies from the relevant literature, conclusions are drawn.

- **Chapter 4: Hybrid particle swarm optimization-butterfly optimization algorithm (PSOBOA)**
  Chapter 4 illustrates a hybrid algorithm to handle constrained optimization problems. It uses a self-adaptation strategy and a parameter-free penalty function with substantial experiments and comparisons with other techniques.

- **Chapter 5: Hybrid chameleon search algorithm-particle swarm optimization (CSAPSO)**
  Chapter 5 discusses image segmentation using a hybrid EA. Several medical image data sets, including COVID-19 chest X-rays, are used to validate performance.

- **Chapter 6: Opposition-based particle swarm optimization-grey wolf optimization algorithm (Opp-PSOGWO)**
  Chapter 6 discusses the modified approach to improving search space exploration and exploitation using the combination of PSO and GWO and Opposition-based Learning. In the end, an analysis of the results and a comparison with other strategies are made.

- **Chapter 7: Conclusion and future scope**
  The final chapter discusses and summarizes the findings of the proposed study along with possible future possibilities.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of relevant studies and research work done

Various EA have been created by modeling the behavior of natural and biological systems. EA have several characteristics, like, Exploration, Exploitation, Diversity, Adaptation, Key Operators, etc. However, the efficiency of EA can be increased using various ways. Some common strategies can be seen from the literature as follows:

2.1.1 Hybridization methods in EA

Hybridization means the fusion of two independent algorithms with their unique set of characteristics to form a better version of the algorithm. Many BFA hybrids were developed to enhance the convergence characteristics and had good results. In (16), fuzzy bacterial foraging was used for harmonic estimation, where the Takagi-Sugeno fuzzy scheme was considered as a cost function. The algorithm worked well in estimation problems that had multimodal landscapes. Khodabakhshian *et al.* in (17) proposed a smart BFA where the tumble was done using a smart unit of length and the new cost function would specify the direction of movement after the tumble. The performance of smart BFA was superior to conventional BFA over different optimization functions. In (18), a hybrid of GA and BFA was developed by incorporating mutation and crossover phenomena in BFA. The algorithm was very effective when tested on benchmark functions, and the application was done on PID parameter tuning.

In (19), a simple hybrid of BFA and PSO was proposed where BFA performed the local search and PSO performed the global search in the search domain. Due to accuracy and good convergence, the hybrid was used to detect bundle branch blocks from ECG patterns. The chemotaxis movement of BFA was incorporated into the ABC exploitation process (20). The hybrid was validated on benchmark functions and was found to be superior compared to its classical counterparts in terms of the convergence rate. DE was hybridized with BFA and named chemotactic DE. The equation of DE was used as the mutation operator in BFA to improve its efficiency (21). The algorithm was applied to calculate the similarity between two indexes. In (22), the exploration ability of spiral dynamics and the exploitation ability of BFA were utilized to make a novel hybrid. The authors had given a simple structure and could achieve better performance on the benchmark models in lesser function evaluations. Gravitational search strategy and swarm search strategy was introduced in BFA to enhance exploration-exploitation ability.

PSO is the most used variant in the hybrid. It is because PSO is simple to implement and has better convergence and good exploitation capability, although it has poor exploration and high chances of local optima stagnation. Hence, the other algorithm used along with PSO in hybrids needs to have a high exploration capability to reduce the chances of local optima traps. Several attempts have been made to develop a hybrid of PSO with other metaheuristic algorithms such as particle swarm optimization with differential evolution (PSODE) (23), particle swarm optimization with genetic algorithm (PSOGA) (24), and Particle Swarm Optimization with Gravitational Search Algorithm (PSOGSA) (25). These algorithms attempt to reduce the local optima trap and achieve the correct balance between exploration and exploitation. These hybrid variants are tested against several benchmark functions and have been proven to perform better than the original PSO. Vikram *et al.* in (26) proposed a hybrid PSOGWO where the position is updated twice, once using the swarm equation and then again using the GWO equation. However, it leads to an increase in complexity and becomes computationally expensive. Similarly, the authors in (27) also proposed a hybrid PSOGWO, where the exploration and exploitation of the search for grey wolves was controlled using the inertia constant. In this algorithm, the exploitation of PSO and the exploration of GWO improved, but the convergence rate increased.

Chopra *et al.* (28) also proposed PSOGWO to solve load dispatch problems, where the GWO first generates all the minimum-valued individuals and they are passed to PSO which then returns the updated values. The drawback of this methodology is that for each iteration, both algorithms run completely to give the result. This leads to instability of the results. In recent studies, analytical approaches and metaheuristic approaches were combined for parameter estimation to outperform conventional algorithms approach (29). However, there remains much scope for validation. A hybrid of bio-inspired and physics-based metaheustic algorithm was developed by researchers in (30), and a through validation was conducted for performance evaluation. It has been shown to be better than other state-of-the-art algorithms and has better convergence. Therefore, there is much scope for more research on this aspect as well. In (31), Golden Jackal and GWO were hybridized to enhance the performance of GWO. It was applied on a high-dimensional dataset for dimensionality reduction and proved good.

2.1.2 Parameter tuning and adaptation methods in EA

In Parameter tuning, tweaking one parameter at a time is done since parameters frequently interact in complex ways. However, the simultaneous tweaking of more parameters results in a huge number of experiments. The following is a summary of the technical issues with parameter tweaking based on experimentation.

i Even if parameters are tuned one at a time, regardless of their interactions, parameter tuning takes time.

ii Although the parameters are not independent of each other, it is nearly impossible to try every potential combination.

iii Despite the tremendous effort put into setting the parameter values, they may not always be the best ones for a given problem.

So, to overcome these problems, Self-Adaptive Parameter Tuning techniques have emerged in recent years. Self-adaptation means modifying the setting of control parameters of the algorithm with each iteration to improve the performance and balance the exploration-exploitation. In terms of the impact of adaptation, the authors in (32) had stated that in the chemotaxis step, if the step size was too large, bacteria would reach the vicinity of the optimum point very quickly and the precision was low. In contrast, if the step size was too small, a lot of steps would be required to reach the convergence, and thus the performance would decrease. So, it was concluded that step size was an important parameter for determining the convergence and error in the output. In this regard, many researchers have explored this area, where the key parameter for adaptation was step size. The principle of adaptive delta modulation was proposed by Tanumay *et al.* (33) to control the step size in BFA. It was successfully applied to the problem of null steering and synthesizing patterns. Majhi *et al.* (34) introduced another version of adaptive BFA where the cost function or the fitness value was used to change the step size value after each iteration. It was successfully applied in a stock prediction model; however, there were a lot of parameters involved in the execution of the algorithm.
In (35), an adaptive step size was introduced based on the mathematical analysis of the fitness function of the bacteria. The main aim of the proposed algorithm was to accelerate the convergence of bacteria to the global optimum value and avoid oscillation around it. It was tested against other state-of-the-art methods and proved effective due to no additional computations in terms of function evaluations. The roulette wheel selection was introduced in the BFA reproduction step along with adaptive change of step size during the chemotaxis movement (36). To enhance the quality of convergence results, serial heterogeneous cooperative search techniques were applied in BFA where different BFA algorithms run sequentially and the output of one algorithm works as the

input for the other algorithm with a smaller step size value (36). The results were much better than those of other comparative algorithms and were proven to be superior.
To enhance exploration-exploitation ability, two chaotic search strategies, namely chaotic initialization and chaotic local search, were incorporated in BFA along with the shrinking strategy (37), and the convergence rate was high in most cases. The quality of the results was enhanced using the gravitational search strategy and the swarm search strategy in the equations of BFA (38). The resultant algorithm was extensively tested on benchmark functions, and the performance was quite good in terms of exploration and exploitation of search space. In (39), the researchers have enhanced the arithmetic optimization algorithm by its conjunction with harris hawk optimizer. It has been able to reduce the local optima trap, however, the time taken for convergence is higher. Recently, hybrid of DE and flower pollination was proposed and it was enhanced using self-adaptation and mutation operator (40). It was tested thoroughly on complex benchmark functions to test its efficiency. However, we observe a limitation of increased time complexity.

2.1.3 Constrained handling methods in EA

A lot of work has been done in recent years to refine the results and develop an efficient algorithm that can solve Constrained Optimization Problems (COP) with quicker convergence. In (41), chaotic maps were used to tune the parameters of GWO and tested on COP. The results were effective in comparison to other algorithms. Trivedi et al. proposed Unified Differential Evolution (UDE) (42) by using the static penalty method within improved DE based local search operation. The problem of the local optima trap was solved to a larger extent, but due to the complexity of the algorithm, the time taken was more. Another variant of PSO was used in case of high dimensional problems, by implementing a dynamic boundary search procedure to handle constraints and proved better than other algorithms (43). Another work in this direction was the Gradient-based Genetic Algorithm (GGA) (44), which exploited the capabilities of gradient-descent algorithms for constraint handling and enhancing the local solutions and gave more favourable starting points in the GA. It gave better results than others in terms of solution precision. In (45), constrained optimization was performed using ABC where the problem was taken as a bi-objective function, one being the objective function itself and the second being the degree of constraint violations. This algorithm was highly effective but the function evaluations were comparatively more so the convergence was a bit slower. So, the well-known algorithms discussed in the literature were able to solve COPs. However, not much was done to effectively explore and exploit the search space. Hence, the scope of better methodology is there which can effectively establish a balance between exploration and exploitation, thereby avoiding local optima trap and also able to give better results with quick convergence. Authors in (46) proposed the solution to constrained optimization problem using improved multiobjective based evolutionary algorithms. The limitation of the method is that the applicability is shown only on limited problems, so its performance on general test problems is still a doubt. In (47), the non-linear constraint optimization problem is solved using double track PSO by identifying the fesing and non-feasible solutions.

2.1.4 Opposition based learning methods in EA

Opposition based learning can prove effective to improve the quality of the solution and increasing the population diversity. Some authors have used OBL along with GWO as well as PSO. In (48), OBL is applied with different hybrid variants of PSO and tested on optimization functions and proved to improve the performance. In Wen Long's "random opposition based GWO" (49), the random opposition is applied to help the population avoid local optima, which is controlled using a C-parameter. In elite-opposition based learning GWO (EOGWO) (50), opposition is done using the limits of the wolves. Apart from OBL alone, a combination of OBL and chaotic technique is also used to further improve the results. Ibrahim *et al.* (51) introduced chaotic opposition based GWO with Differential Evolution, where the GWO was first enhanced using chaotic opposition learning and then updated repeatedly using Differential Evolution operators. This algorithm is tested against several benchmark functions and gives significant results. OBL and GWO are applied together in task area as well (52).

2.1.5 Multilevel image thresholding in medical imaging applications using EA

Researchers in the medical domain have used multi-level thresholding with nature-inspired algorithms in diverse ways. In a study (53), researchers proposed a novel method of updating the context vector during the cooperation process in a dynamic way where they used Otsu as an objective function. The testing was done on the CT scans of the stomach and was found effective. However, the computational complexity of the method may limit its scalability to large datasets or real-time applications. In 2012, three-level thresholding of human head CT images was done using improved Biogeography-Based Optimization (BBO) technique, where the fuzzy entropy was the objective measure to be maximized (54). The comparison was with original BBO, PSO, and other state-of-the-art methods. The improved BBO proved superior and more flexible than its original algorithm and other methods. However, a careful selection of parameters is required. Abbas *et al.* (55) proposed a refined segmentation method based on pixel-probability density function and edge detection. The technique was applied to High-Resolution Computed Tomography (HRCT) of lungs, and compared with other segmentation methods. Brain Magnetic Resonance (MR) has been a targeted area for many researchers. However, it lacked generalization and validation on other datasets. In (56), the author proposed multi-level thresholding of brain MR images by computing the 2-d histogram-based grey gradient to preserve the edges of the brain, using Shallow Swarm Optimization (SSO). The comparisons were done with other soft computing approaches. The complexity of the algorithm was high, although it was effective in performance and gave good results. But, the proposed method is not compared to the state-of-the-art segmentation techniques to demonstrate its superiority. Also, in (57), Brain MR image segmentation was done using an enhanced Shuffled Frog Leaping Algorithm. It gave good results and was qualitatively and quantitatively better than other competitive approaches but the sample size used was very small so effectiveness cannot be claimed. BA was used in tsallis entropy-based thresholding for image segmentation (58). It used the region growing segmentation method and extracted the texture features to final abnormalities in MR images of the brain and breast. Li *et al.* used partitioned and cooperative quantum-behaved PSO to optimize the Otsu thresholding parameters and applied them to optimize complex benchmark functions and four stomach CT images (59). The results were better in terms of inter-class variance. The approach has a high computational cost, which could make it challenging to use in real-time or in circumstances where resources are few. The dataset used was limited and the algorithm lacked analysis of the parameters. Wang *et al.* (60) used Otsu as the objective function and tried optimizing the thresholds using improved Flower Pollination Algorithm with a random location operator. The performance was evaluated using gray-scale medical images and gave robust and effective output as compared to other algorithms.

Several multi-level thresholding techniques have been proposed so far that employ different criteria functions, such as Otsu's between-class variance (61), minimum error thresholding (62), and entropy-based thresholding (63). Researchers have explored all of these methods over the last decades. However, entropy-based thresholding has gained much popularity. Some common entropy measures used as objective functions include Kapur's entropy (64)(65), fuzzy entropy (66), cross-entropy (67), Tsallis entropy (68), Shannon entropy (69), and Masi entropy (70). Among these, the Minimum Cross-Entropy Thresholding (MCET) function is widely utilized. An optimal threshold value segments the image based on the difference between the original image and the ideal segmented image. The challenge of increased complexity still prevails when performing multi-level thresholding. To address this, researchers have incorporated metaheuristic algorithms to enhance threshold accuracy. In (71), bi-level and multi-level image segmentation was done by maximizing Otsu's between-class variance, using the firefly algorithm, Brownian distribution, levy flight, and the Gaussian distribution. Color image segmentation based on multi-level MCET using the cuckoo search algorithm was used in (72), where authors compared the proposed approach with other metaheuristic-based

image segmentation techniques, and the proposed approach proved highly effective. For complex image analysis, PSO was incorporated into MCET to find the optimal thresholds during multi-level image segmentation (73). Another metaheuristic called grey wolf optimizer was applied to thresholding problems with Otsu and Kapur's entropy as its objective function (74). After an extensive experimental process, this approach proved to be better than thresholding done using PSO and BFA. In (75), DE was applied to Tsallis-fuzzy entropy-based image thresholding, showing statistical significance over other state-of-the-art methods using Shannon entropy or fuzzy entropy.
These studies highlight the advancements in utilizing metaheuristic algorithms to improve the accuracy and effectiveness of multi-level thresholding in image segmentation. However, there is always a limitation where some optimization algorithms are good in exploring the search space and some are good in exploitation. Several hybrid metaheuristics are proposed in the related field to strike an exploration-exploitation balance, avoid premature convergence, and have shown superior performance. A hybrid GSA and the GA were used in multi-level thresholding where between-class variance and entropy functions were used as fitness functions. In this approach, a novel method of evaluating the standard deviation after fitness function calculation was employed to ensure diversity (76). Otsu thresholding was optimized using a hybrid firefly algorithm and social spider optimization, where processing time was significantly reduced, along with good results (77). In (78), Ewees *et al.* gave a hybrid of ABC and the Sine Cosine Algorithm (SCA), where SCA improved the local search ability of ABC to enhance its performance in image thresholding. It is imperative to note that hybrid optimization algorithms have shown better performance than individual optimization algorithms.
Firefly Algorithm (FA) was used to segment lung CT images (79) and brain MRI (80). The paper proposes emphysema classification in lung tissue CT images using Support Vector Machine (SVM) optimized by the FA. The proposed approach's computational time and complexity are not fully analysed in the research, which is crucial for real-world applications. The GWO was used in an artificial neural network for the classification of the MRI images dataset (81). The hybrid approach of using grey wolf optimizer and artificial neural network improves the accuracy of classification for MR brain images. But, the scope of the algorithm is limited incase of general test problems. Moreover, Soham *et al.* (82) gave the concept of multi-level thresholding using a multi-objective evolutionary algorithm where two entropies namely, minimum cross-entropy and Renyi entropy, were used as the objective function. It is an effective technique to segment brain tumors. In this regard, automatic segmentation of MR images was done using Crow Search Algorithm (83). This method used minimum cross entropy as its objective function. The performance was evaluated on general as well as medical images. Ultrasound image segmentation was carried out using a multi-level Otsu thresholding

method optimized using Differential Evolution Algorithm (84). The proposed differential search algorithm-based method achieves higher segmentation accuracy compared to other methods, as demonstrated by experiments on real ultrasound images.
A detailed comparison of different optimization techniques in the field of multilevel thresholding is discussed in (85). For the analysis and classification of diseases in healthcare systems, machine learning and data mining approaches have become increasingly popular in recent years. In (86), authors give a review of various machine learning methods used in health care, the challenges and the possible solutions. In a manner similar, (87) investigated sub-features for classification in data mining, which was helpful for enhancing the precision of disease classification models. Analytic methods for e-Health data have also been deployed using soft computing and machine learning approaches. (88) suggested a framework for applying these methods to healthcare data analysis and patient outcomes improvement. A machine learning and medical stuff model was also used by (89) to analyse classification-based projected disease, which could be useful for diagnosing and forecasting diseases in patients. However, the sample size used was small.
We noticed a significant gap in the literature in the above-mentioned areas, and we identified the need for a dataset of sufficient size to address this issue. Additionally, we found that previous studies lacked proper descriptions of the parameters used for tuning and measures of accuracy.

CHAPTER 3

HYBRID BACTERIAL FORAGING ALGORITHM-FIREFLY ALGORITHM (BFAFA)

Global optimization involves finding the global optimal values for the objective functions that contain local optima. The optimization algorithms search the entire input space and get closer to the extrema of the function to give the accurate value. Among these optimization algorithms, swarm-based algorithms have gained a lot of importance over the years in solving global optimization problems. They are extensively used to solve engineering and scientific problems. Some swarm-based algorithms are PSO (2), BFA (1), SSA (4), etc.

3.1 Introduction

Among the bio-inspired optimization algorithms, BFA is based on the Escherichia Coli (E. Coli) bacteria's foraging behavior i.e., the food-seeking and reproductive behavior. The bacteria search in the nutrient medium to increase the energy per unit of time. It is a random searching algorithm that finds the global optimum value using the process of chemotaxis, reproduction, elimination, and dispersion. In the computational community, BFA has received a great deal of attention due to its exceptional performance in finding global optima. It was successfully applied in optimal control (90), machine learning (91), transmission loss reduction (92), harmonic estimation (93), multi-objective optimization problems (94), image enhancement (95) and edge detection (96)(97)(98)(99), optimal multilevel thresholding (100), vehicle routing problems (101), numerical optimization (102), optimal placement of distributed generation (103), and so many other areas. However, BFA has poor convergence as compared to other swarm-based algorithms over multimodal and rough landscape problems (104). This is because they follow the local search through the chemotaxis movement. On the other hand, the firefly algorithm is a successful nature-inspiring optimization algorithm. In this algorithm, there are three main rules; 1) Fireflies get attracted to each other, independent of their sex, 2) Attractiveness is proportional to Brightness and vice-versa, and for any two fireflies, the movement takes place from less bright firefly to brighter firefly.

If no firefly is brighter than a specific Firefly, it will randomly travel in any direction, 3) The brightness of a Firefly is evaluated by the objective function.

To improve the performance of BFA, a lot of modifications have already been made, which can be broadly categorized as Hybridization and Self-Adaptation. Hybridization means the fusing of two independent algorithms with their unique set of characteristics to form a better version of the algorithm. Self-adaptation means the setting of control parameters of the algorithm with each iteration to improve the performance and balance exploration-exploitation.

Although the literature mentioned above had improved BFA, they were effective in specific problems and lacked universality. As observed, there were still many open problems that needed to be addressed. They can be defined as follows.

i It was learned from the literature that some algorithms involved too many control parameters to balance exploration-exploitation, which made the problem more complex. There was a need for improving the convergence using fewer parameters and achieving effective parameter-tuning based on adaptation. Until now, all adaptation strategies are based only on the fitness values of the bacteria or the iteration number.

ii In multimodal or any other complex landscape where the search space is rough and has multiple local optima, the algorithm got stuck in the local optima trap and gave poor results.

iii From the related works, it was observed that mostly, very few evaluation parameters such as convergence rate, multidimensional landscape, Wilcoxon rank test, etc. were used to prove the superiority of the algorithm. There was a dire need for extensive analysis of the algorithm with other comparative algorithms to prove the quality of the results.

In addition, Wolpert et al. (105) had mentioned the no free lunch theorem where no algorithm was better on all problems, and so there was always a scope for a new algorithm that outperforms other algorithms. These points are the motivation of this work to optimize real-world problems. Unlike other works mentioned in the literature, this study proposes adaptation of the step size based on the success rate of the swarm. Furthermore, the leadership strategy is proposed in BFA for the first time to update the positions of weak bacteria and move toward a strong bacterium.

The main aim of this chapter is to improve the BFA for solving real-world problems. For this purpose, two novel improvements are proposed in conventional BFA, namely adaptation and hybridization. In adaptation, the dynamic step size is proposed using the success percentage of the swarm of bacteria. A higher success value means that the swarm is moving closer to the optimum, whereas a lower value of success means the swarm is far from the optimum. This step prevents the user-based initialization of key parameters; hence, the algorithm is free from parameter-tuning. It leads to better exploration-exploitation balance, prevents local optima traps, and works well even on complex optimization problems. In hybridization, the leadership strategy is proposed where the bacterium with the least fitness value (most healthy) is the leader and the other bacteria in the swarm follow the leader using the FA position equation. This leadership approach is used instead of the elimination and dispersion step. In this way, the weak bacteria are not eliminated, rather they are moved to a position of a high food source that is close to the best bacteria resulting in the rejuvenation of the population of bacteria. FA (3) has a fast convergence rate and involves fewer computations; hence it is most appropriate to hybridize FA with BFA and improve its convergence speed. It is interesting and promising to study the capability of leadership strategy and the success percentage of the swarm in optimizing results. BFAFA is tested against classical, single-peak, multi-peak, and noisy nonlinear benchmark functions and CEC_2017 benchmark functions and compared to other well-established and recent metaheuristics. Scalability analysis is done by testing the algorithm in a multidimensional environment. The significance of the algorithm is validated using the Wilcoxon rank sum and Friedman rank tests. The results are verified by addressing the two classical engineering problems i.e. Cantilever beam design and three-bar truss design.

3.2 Strategies employed in the BFAFA algorithm

The proposed algorithm introduces two novel improvements in BFA, namely adaptation and hybridization, to improve convergence and accuracy, avoid local optima, and free the algorithm from parameter tuning.

3.2.1 Adaptation

In the chemotaxis step of classical BFA, step-size C is a constant parameter and determines the amount of movement. However, it is observed that if there is a dynamic change in step size the convergence is affected greatly. In this chapter, we dynamically determine the chemotactic step size during a run, to strike a balance in their exploration and exploitation behavior. A bacterium with a small step size has the capability of exploitation, and one with a large step size has the capability of exploration. The large value of the step size accelerates the movement of the bacteria towards the optimal point, while the small value of step size slows the bacteria to converge to the optimal point.

To achieve the adaptation, the situation of the swarm is first determined at each iteration for which the success percentage is introduced in this chapter. If the success percentage is high, it means the swarm of bacteria is converging towards the global optimum. On the contrary, the low value of success percentage means there is not much improvement

in the bacteria.
The steps of adaptation are as follows:

1. The success of p^{th} bacteria at the q^{th} chemotactic step and r^{th} reproduction step is defined in a minimization problem according to Eq. (3.1).

$$success\_count(p) = \begin{cases} 1 & if F(p,q,r) < F_{last} \\ 0 & if F(p,q,r) \geq F_{last} \end{cases} \tag{3.1}$$

   where $F(p,q,r)$ is the fitness function to be optimized and F_{last} is the best fitness value of the bacteria obtained so far.

2. The success percentage of the bacteria using the success values is computed according to Eq. (3.2):

$$PS(p) = \frac{\sum_{p=1}^{s} success\_count(p)}{s} \tag{3.2}$$

   where s is the total number of bacteria, and PS is the percentage of p^{th} bacteria that had an improvement in their fitness value during the last iteration.

3. A linear function is used to map PS values to the possible range of step size as shown in Eq. (3.3)

$$C = (C_{max} - C_{min})PS(p) + C_{min} \tag{3.3}$$

   To determine the range of C, multiple values between 0 and 1 were tried experimentally. It was found that the values of C between 0.2 to 0.5 give good results. So the lower range of C, C_{min} is taken to be 0.2 and the upper range C_{max} to be 0.5.

In this way, the step size is regulated and adaptively changed from high value (exploration) to low value (exploitation) to reach the global optimum. This ensures the best possible exploration-exploitation balance. This way, user interference is avoided to initialize the key parameter; hence the algorithm is free from key parameter-tuning. Since the situation of the swarm is considered at every step, and not the single bacterium, therefore, the chances of local optima trap are reduced significantly. This makes it perform better in rough landscapes also, like in the case of multimodal optimization algorithms. To the best of our knowledge, the success rate of the swarm of bacteria to determine the step size is used for the first time in BFA.

3.2.2 Hybridization

In the conventional elimination and dispersion stage, every bacterium is mapped to a random value between 0 and 1. If the random value is less than N_{eldis}, the bacterium is eliminated. An equal number of fresh bacteria are introduced at random locations to maintain the total count. However, in this event, the biological diversity is lost and a potential optimal bacterium can be killed. This chapter proposes a new strategy to overcome this situation. The standard elimination and dispersion step is avoided altogether for this. To achieve this, the steps are as follows:

1. The bacteria are sorted in descending order, according to their health. The health of bacteria is calculated using the below code.
   For $p = 1, 2, ..s$
   $J^p_{health} = \sum_{j=1}^{N_c+1} F(p, q, r)$ be the health of bacterium p
   end for
   Sort bacteria with J_{health} in descending order

2. The bacterium with the least fitness value is considered most healthy (for minimization problems) and is termed as Lead.
   Lead=Last bacterium of the sequence

3. The positions of the weaker set of bacteria (first half of the sorted population) are then modified using the position update equation of the FA, where the weaker bacteria move towards the strongest bacterium.
   For $i = 1, 2 \ldots, \frac{s}{2}$
   $pos(i, q+1, r) = pos(i, q+1, r) + \beta_0 e^{-\gamma dist^2_{iLead}}$
   $(pos(i, q+1, r) - Lead) + \alpha(rand - \frac{1}{2})$
   end For
   where $dist_{iLead}$ is the distance between i^{th} bacterium and Lead bacterium and is calculated using Eq. (1.6); N_c is the number of chemotactic steps.

By introducing the randomness in the position of weak bacteria, we diversify the search space. In this way, exploration is improved several times. So, rather than killing the weak bacteria, an attempt to make them closer to nutrients is done by changing their position and rejuvenating them. Since we know that fireflies change position when they get attracted to brighter firefly, in our case, the weak bacteria (followers) are attracted to the strongest bacterium (Leader), which is most close to the nutrient medium, and update their positions accordingly. The strategy of leadership to find the best food resource works excellent (106)(107). In this way, search efficiency (or convergence) is improved greatly, and diversity is also achieved leading to better exploration. The chances of missing the global optimal solution are reduced. As compared to other nature-inspired algorithms, FA gives quick convergence in fewer computations, hence it is most suitable to fuse with BFA, reduce the total runtime, and give good results with quick convergence. From the literature, it is seen that other swarm-based optimization algorithms were used for hybridization with BFA; however, FA is used for hybridization with BFA for the first time.

3.3 Development of the BFAFA algorithm

The overall structure of the novel BFAFA algorithm with adaptation and hybridization is shown in Algorithm 6. The population is randomly initialized and all the parameters are set. During the chemotaxis movement (swim or tumble), the step size is dynamically adjusted according to the success percentage of the swarm. The bacteria move ahead and the new fitness value of the entire swarm is evaluated. If the current fitness value of the bacteria at the new position is smaller than the previous value, the bacteria update its position. For reproduction process, the health of the bacterium is determined. In the proposed algorithm, the elimination and dispersion step is not used. So, the bacterium with the best health (least value of fitness), which is determined in the reproduction step, is defined as the lead, and the rest of the bacteria become the followers and update their position according to the firefly equation and move towards the healthy bacteria. So, the overall number of bacteria remains constant. Once the bacteria move closer to the lead, the health (fitness value) of the bacteria is improved. In this way, potentially strong bacteria are not eliminated and biological diversity is increased. The overall health of the swarm is also improvised.
By encouraging constant fitness value improvement as bacteria converge towards the lead, this novel strategy maintains diversification, strengthens potentially robust bacteria, and improves the general health of the swarm. Moreover, the lack of processes for dispersal and elimination shields potentially potent bacteria from early extinction, promoting long-term diversity. This unique adaptability, in line with dynamics inspired by fireflies, guarantees a balance between optimizing the health of each individual and the entire swarm. The resulting approach emphasizes dynamic adaptation and robustness, providing an innovative perspective on swarm-based optimization.

Algorithm 6 The proposed BFAFA algorithm

```
1. Parameter definition: BFA parameters: $D$: the dimension of search space;
   $s$: population of bacteria; $N_c$: number of chemotactic steps; $N_s$: number of
   swims after tumble; $N_{re}$: number of reproductive steps; $pos$: position of each
   member in the population of s bacteria; $C$: step size; $C_{min}$: minimum step
   size; $C_{max}$: maximum step size; $PS$ : Success percentage. FA parameters: $\alpha$:
   randomization parameter; $\beta$: attractiveness; $\gamma$: light absorption coefficient
```

```
2. Initialize all the parameters defined above
3. $success\_count = 0$, $PS = 0$
4. Reproduction Loop: while $r < N_{re}$
5. Chemotaxis Loop: while $q < N_c$
    (a) $PS = \frac{success\_count}{s}$
    (b) $C = (C_{max} - C_{min})PS + C_{min}$
    (c) for $p = 1, \ldots, s$
    (d) Compute Fitness cost, $F(p, q, r)$
    (e) $F(p, q, r) = F(p, q, r) + J_{cc}(pos)$, where $J_{cc}(pos)$ is calculated using
        (1.3)
    (f) $F_{last} = F(p, q, r)$
    (g) Tumble: Generation of a random vector $\Delta(p) \epsilon R^D$ with each element
        $\Delta_d(p), d = 1, 2, ...D$, a random number $[-1, 1]$ where $R$ is a real num-
        ber.Computing $pos$ for $p = 1, 2 \ldots, s$ (number of bacteria) $pos(p, q +
        1, r) = pos(p, q, r) + C\frac{\Delta(p)}{\sqrt{\Delta^T(p)\Delta(p)}}$
    (h) Swim: Set swim length counter, $m = 0$
    (i) While $m < N_s$
    (j) If $F(p, q + 1, r) < F_{last}$ (if there is any improvement)
    (k) $success\_count = success\_count + 1$
    (l) $PS = \frac{success\_count}{s}$
    (m) $C = (C_{max} - C_{min})PS + C_{min}$
    (n) let $F_{last} = F(p, q + 1, r)$
    (o) $pos(p, q + 1, r) = pos(p, q, r) + C\frac{\Delta(p)}{\sqrt{\Delta^T(p)\Delta(p)}}$
    (p) Use $pos(p, q + 1, r)$ to calculate the new fitness function $F(p, q, r)$
    (q) else
    (r) $m = N_s$
    (s) End if
    (t) $m = m + 1$
    (u) End While
    (v) End for
6. End while (Chemotaxis loop ends here)
7. Reproduction: For given $r$, and For $p = 1, \ldots, s$
    (a) $J^p_{health} = \sum_{j=1}^{N_c+1} F(p, q, r)$ be the health of bacterium $p$
    (b) end for
    (c) Sort bacteria with $J_{health}$ in descending order
    (d) Select Lead=Last bacterium of the sequence
    (e) computing new positions of follower bacteria using the firefly algorithm
        for $i = 1, 2 \ldots, \frac{s}{2}$
        $pos(i, q+1, r) = pos(i, q+1, r) + \beta_0 e^{-\gamma dist^2_{iLead}}(pos(i, q+1, r) - Lead) +
        \alpha(rand - \frac{1}{2})$
    (f) end for
8. End while (Reproduction loop ends here)
```

3.4 Experimental results and analysis

3.4.1 Evaluation methodology

An experimental analysis of the proposed approach is done using MATLAB R2010a on 64-bit Windows to validate its performance. Classic benchmark functions, including unimodal, multimodal, and composite functions, are used from (108) to test the proposed algorithm. Along with standard benchmark functions, the experimental analysis is performed on single-peaks, multi-peaks (109), noisy nonlinear benchmark functions (110) and modern numerical optimization problems from IEEE CEC_2017 (111)(112). The CEC_2017 functions have features such as shifting, rotating, and expanding the basic functions or a variant of these. To verify the results of the proposed BFAFA algorithm, it is compared with some well-established and recent metaheuristic algorithms. The well-established algorithms include BFA (1), FA (3), ABC (113), and the recent algorithms used are Monarch Butterfly Optimization (MBO) (114), SSA (4), Hybrid BFA and PSO (BF-PSO) (115). In all the experiments, the population size is set to 50, the number of iterations to 100, and the evaluation is taken for 30 different runs. The parameters of the proposed approach are set as follows: $s = 50$; $N_c = 100$; $N_s = 4$; $N_{re} = 4$; $C_{min} = 0.2$; $C_{max} = 0.5$; $\gamma = 1$;$\beta_0 = 1$; $\alpha = 1$. The rest of the parameters of the other algorithms used in the comparison are the same as those used in the literature. Performance is evaluated as follows:

- Qualitative Analysis – Analyzing the convergence curves of the proposed method and other algorithms.
- Quantitative Analysis Validating the proposed method against other comparing algorithms by evaluating the fitness function over classic, noisy nonlinear, single peak, multi-peak, and modern CEC benchmark functions, comparing the time complexity, performing the scalability analysis, and performing Wilcoxon statistical analysis at a 5% significance level, and Friedman rank test.

3.4.2 Qualitative results and discussion

The first set of experiments is performed by observing the Convergence Curve of the proposed approach and comparing it with the convergence curves of other metaheuristic algorithms. The main aim of conducting this experiment is to observe the behavior of BFAFA qualitatively. The benchmark functions used are listed in Table 3.1, where Dim shows the dimension of the function, the range is the limit of the search space, and the value f_{min} gives the optimal value.

Table 3.1: List of benchmark functions used for optimization

Type of Function	Function	Dim	Range	f_{min} value
	$f_1(x) = \sum_{i=1}^{s} x_i^2$	30	[-100,100]	0
	$f_2(x) = \sum_{i=1}^{s} \lvert x_i \rvert + \prod_{i=1}^{s} \lvert x_i \rvert$	30	[-10,10]	0
Unimodal benchmark functions	$f_3(x) = \sum_{i=1}^{s} (\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	0
	$f_4(x) = \max_{i=1,\ldots,s} \lvert x_i \rvert$	30	[-100,100]	0
	$f_5(x) = \sum_{i=1}^{s} ([x_i + 0.5])^2$	30	[-100,100]	0
	$f_6(x) = 418.9829d - \sum_{i=1}^{s} x_i sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	-418.9829 X 5
	$f_7(x) = 10n + \sum_{i=1}^{s} (x_i^2 - 10cos(2\pi x_i))$	30	[-5.12,5.12]	0
Multimodal benchmark functions	$f_8(x) = -20.exp(-0.2\sqrt{\frac{1}{s}\sum_{i=1}^{s} x_i^2}) - exp(\frac{1}{s}\sum_{i=1}^{s} cos(2\pi x_i)) + 20 + exp(1)$	30	[-32,32]	0
	$f_9 = \frac{\pi}{s}\{10\sin^2(\pi y_1) + \sum_{i=1}^{s-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_s - 1)^2\} + \sum_{i=1}^{s} u_i[1cm]\lvert x \rvert \leq 50$ $u_i = \begin{cases} k(x_i - a)^m[1cm]x_i > a \\ 0[2cm] - a \leq x_i \leq a \\ k(-x_i - a)^m[1cm]x_i < -a \end{cases}$ $y_i = 1 + (x_i + 1)/4$ $x* = [1, \ldots 1]$ $f(x*) = 0$	30	[-50,50]	0
	$f_{10} = 0.1\{\sin^2(\pi 3x_1) + \sum_{i=1}^{s-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_s - 1)^2[1 + \sin^2(2\pi x_s)]\} + \sum_{i=1}^{s} u_i[1cm]\lvert x_i \rvert \leq 50$ $u_i = \begin{cases} k(x_i - a)^m[1cm]x_i > a \\ 0[2cm] - a \leq x_i \leq a \\ k(-x_i - a)^m[1cm]x_i < -a \end{cases}$ $x* = [1, \ldots 1]$ $f(x*) = 0$	30	[-50,50]	0
	$f_{11}(x) = exp(-\sum_{i=1}^{s}(x_i/\beta)^{2m}) - 2exp(-\sum_{i=1}^{s} x_i^2)\prod_{i=1}^{s} cos^2(x_i)$	30	[-20,20]	-1
fixed dimension multimodal benchmark function	$f_{12}(x) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)$			
	$[30 + (2x - 3y)^2(18 - 32x + 12x^2 + 4y - 36xy + 27y^2)]$	2	[-2,2]	3

Fig. 3.1 shows the 2D versions of the benchmark functions that are used in this chapter for testing purposes. These minimization functions can be simply categorized as unimodal, multimodal, and multimodal (fixed dimension). Studying the pattern of the proposed algorithm on these functions helps to generalize the results for other unimodal and multimodal functions as well. Exploitation can be benchmarked using unimodal benchmark functions (f_1 to f_5). In contrast, multimodal functions (f_6 to f_{11}) have a lot of local optima, which can be used to benchmark exploration. From f_{12}, which is a composite function, the capability of exploration and exploitation can be simultaneously benchmarked, along with the capability of local optima avoidance. The convergence curves of the proposed algorithm and various other metaheuristic algorithms are depicted in Fig. 3.2, for all the benchmark functions. From the convergence curves, it is clear that BFAFA performs well over other metaheuristic algorithms in terms of convergence speed. For most of the unimodal functions (except f_3), BFAFA shows significant results which means that they have merit in exploitation. As per the curves for the function f_7 to f_{11} (except f_6), BFAFA gave superior results which makes it capable of good exploration, although BF-PSO gave competitive performance. It can be derived that BFAFA performs well on multimodal optimization functions and, at the same time ensures quick convergence. The proposed approach is fairly competitive to other algorithms in terms of convergence speed for composite functions as well, showing a good balance between exploration and exploitation. This shows that it balances exploration-exploitation effectively to drive the bacteria to the global optima. From the convergence trend, it is further noticeable that BFAFA is better than its counterparts BFA and FA. The convergence of BFA is superior to FA, however, the convergence of BFAFA outperforms both of them. In most cases, convergence is slow at the initial iterations in the exploration phase, however, it picks up the speed and converges faster at a later stage. This is possible due to adaptation which ensures efficiency in search space during exploration, and acceleration in convergence later. The fusion of FA into BFA assists in ensuring fast convergence in a few computational steps, so it contributes to increasing the convergence rate, which is validated in the quantitative analysis section.

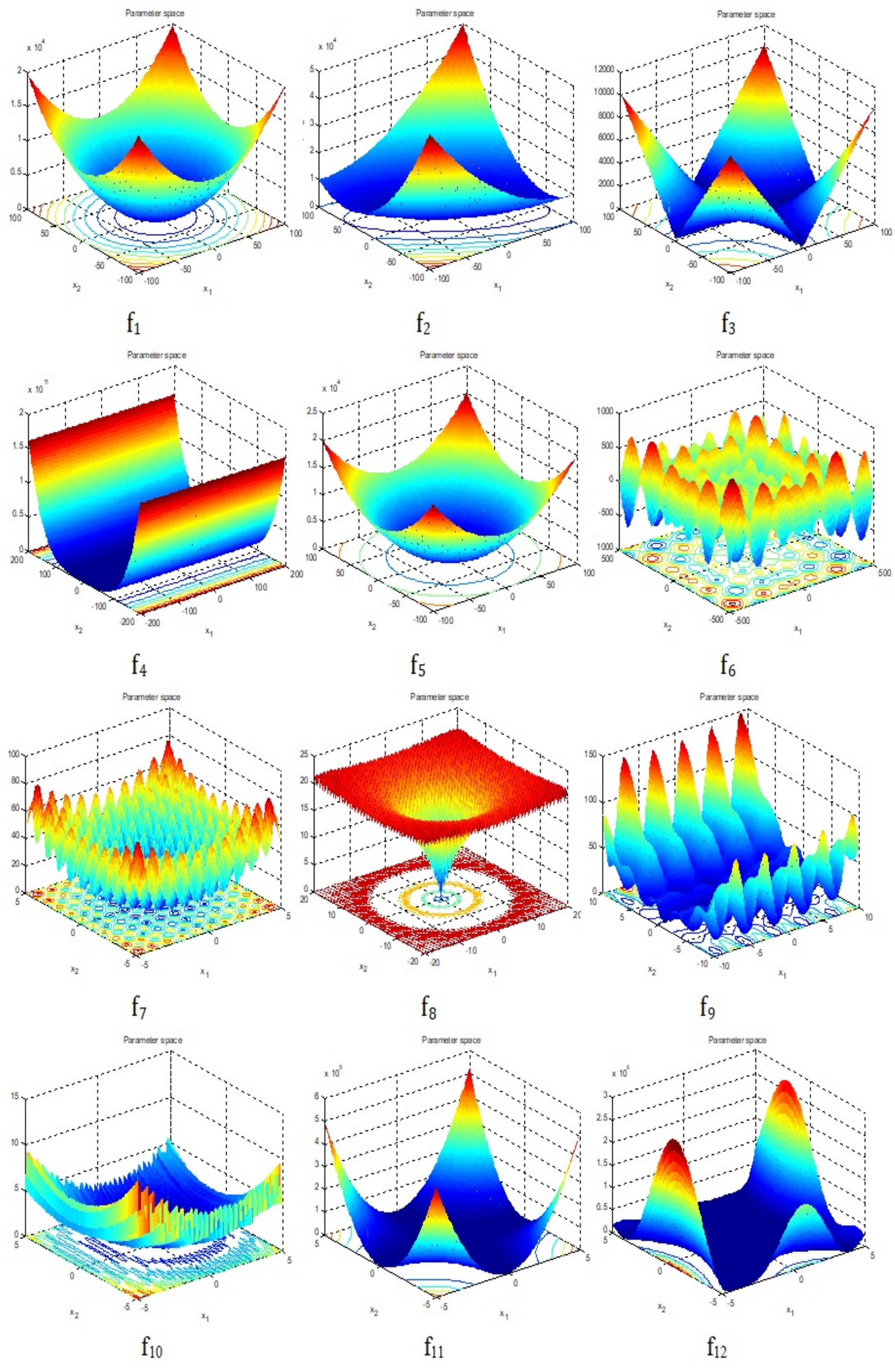
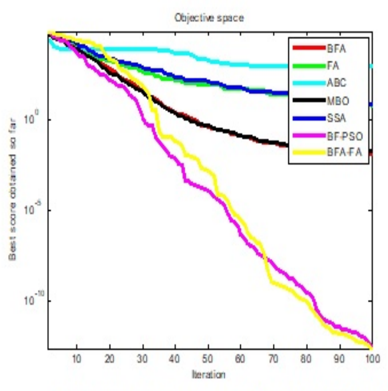
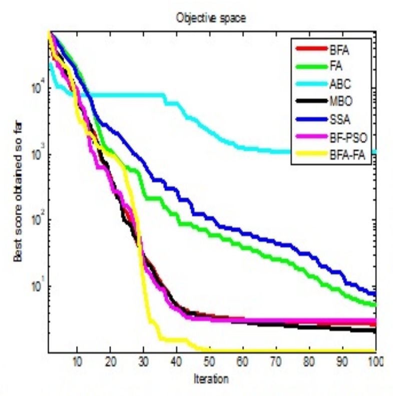


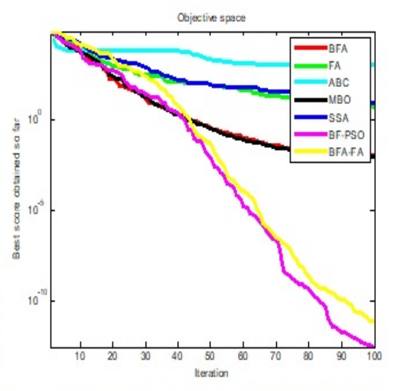
Figure 3.1: 2-D versions of the unimodal and multimodal benchmark functions



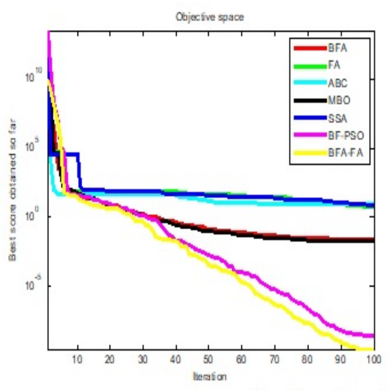
convergence curves for function f_1



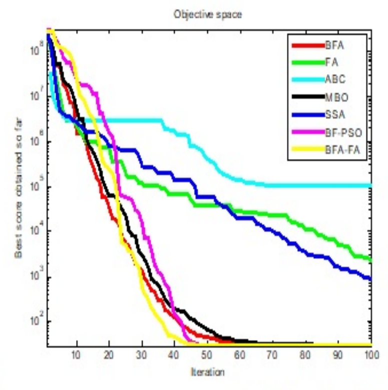
convergence curves for function f_2



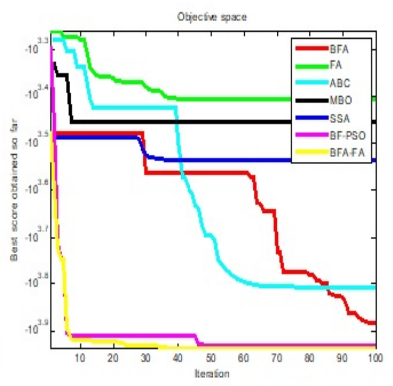
convergence curves for function f_3



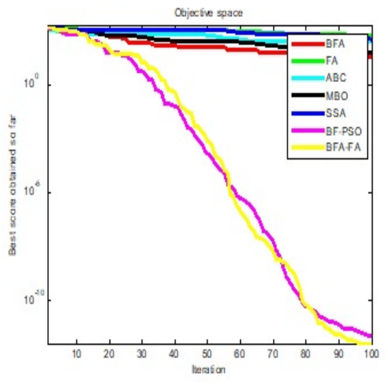
convergence curves for function f_4



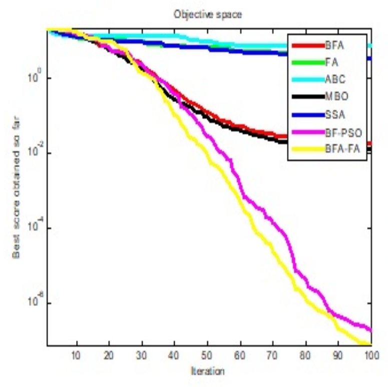
convergence curves for function f_5



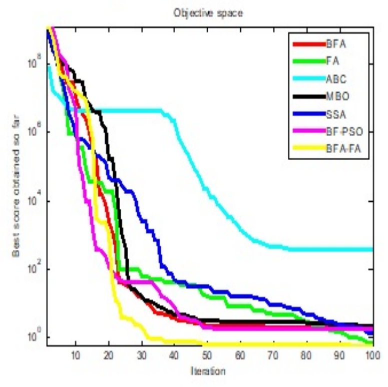
convergence curves for function f_6



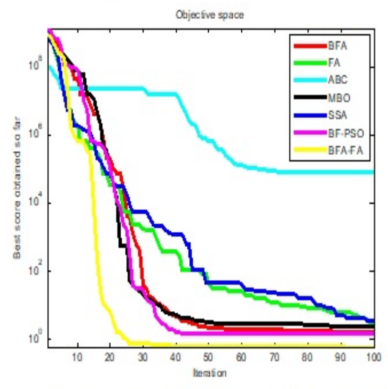
convergence curves for function f_7



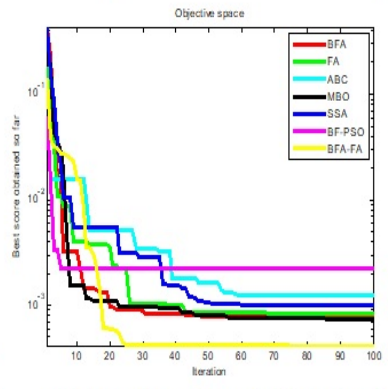
convergence curves for function f_8



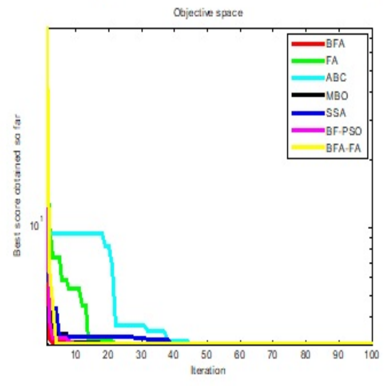
convergence curves for function f_9



convergence curves for function f_{10}



convergence curves for function f_{11}



convergence curves for function f_{12}

Figure 3.2: Convergence curves of the proposed BFAFA and other algorithms on the benchmark functions

Table 3.2: Fitness values of functions using a different range of step size

Range of step size C		f_1	f_8	f_{12}	C_1	NL_1
C_{min}	C_{max}					
0.1	0.3	0.0055e+00	3.6200e-11	3.2400e+00	1.8892e-15	1.982
0.2	**0.5**	**8.8818e-16**	**0.0000e+00**	**3.0000e+00**	**0.0000e+00**	**1.4789**
0.3	0.8	4.8743e-02	1.59239e-01	3.6183e+00	5.3847e-20	12.9137
0.4	0.9	4.8021e-01	1.2365e-01	3.6900e+00	3.8473e-44	13.8373

3.4.3 Quantitative results and discussion

The second experiment was based on evaluating the fitness value, mean, standard deviation, root mean square, mean absolute error, mean absolute percentage error, normalized root mean square error, and convergence time obtained by the proposed algorithm and comparing it with BFA, FA, ABC, MBO, SSA, Hybrid BF-PSO.
To achieve the best results for any optimization problem, the control parameters of the algorithm should be tuned appropriately. This could provide more robustness and adaptability when solving a general set of problems. For our algorithm, the range of step-size C was selected using the Design of Experiments approach. The sensitivity of the algorithm was analyzed by trying various ranges of C between 0 and 1 and then performing the experiment. The values varied until a decent solution was obtained. The analysis was done on benchmark functions f_1, f_8, f_{12}, CEC_2017 function C_1, noisy nonlinear function NL_1. Each function was solved for 30 different runs. The results of some of the ranges of step sizes are shown in Table 3.2. From the sensitivity analysis done in Table 3.2, it is evident that too small or too large a difference between the range of step size gives inaccurate results, and the performance is much better for a reasonable range when tested for 30 independent runs. Hence C_{min} was taken as 0.2 and C_{max} was taken as 0.5.
Table 3.3 shows the evaluation results of optimal values for different algorithms. The best results of the global optima are highlighted. The evaluation parameter here was the fitness function values obtained by the various algorithms over 30 different runs. As discussed, the functions f_1 to f_5 benchmarks exploitation, it can be observed from the table that BFAFA outperformed all comparative algorithms in almost all the functions except f_5. The results for the function f_5 were slightly better given by BFA and MBO. For the function f_1, BFAFA as well as BF-PSO gave the same results which shows that these two algorithms are competitive. It can therefore be generalized that

Table 3.3: Fitness function f(.) evaluation on various optimization functions

Function	Optimum Value, f(.)						
	BFA	FA	ABC	BF-PSO	MBO	SSA	Proposed BFAFA
f1	0.4021e+00	0.0066e+00	0.0065e+00	8.8818e-16	5.7119e-05	4.8613e-06	8.8818e-16
f2	1.9645 e+00	8.1553e-04	0.9986e+00	8.8818e-16	3.6200e-11	5.4306e-07	0.0000e+00
f3	1.4950e-09	9.8998e-06	1.0276e-05	1.3298e-32	5.6500e-09	1.8418e-14	1.4998e-34
f4	0.1471e+00	3.0002e-02	1.0044e-07	5.0339e-04	3.0012e-08	1.0018e-03	3.0076e-09
f5	0.0000e+00	1.5393e-04	0.0122e+00	1.4998e-32	0.0000e+00	3.8327e-04	0.0067e+00
f6	-1.1260e+03	-5.8275e+07	-7.2783e+04	-4.0327e+03	-7.3423e+03	-5.4306e+04	-6.0024e+03
f7	0.1100e+00	7.8730e-06	2.6951e-05	0.0000e+00	0.0000e+00	3.2110e-14	0.0000e+00
f8	0.1680e+00	4.7375e-07	4.2123e-08	0.0000e+00	0.0000e+00	3.1326e-07	0.0000e+00
f9	-1.0020e+00	2.5908e-06	9.1240e-06	8.8818e-16	4.1240e-11	3.2110e-14	0.0000e+00
f10	1.3584e-02	0.9999e-07	0.8110e-06	8.8395e-17	-1.0000e+00	8.7289e-01	8.8261e-30
f11	-1.8074e+00	-1.6633e+00	-1.3860e+00	-1.5660e+00	–1.7309e+00	-1.5819e+00	-1.0230e+00
f12	3.0300e+00	3.9285e+00	3.3700e+00	6.3890e+00	3.8013e+00	3.6400e+00	3.0000e+00

BFAFA can solve unimodal optimization functions with nearly accurate global optimum values. So, the proposed algorithm exhibits good exploitation capability. The functions f_7-f_{11} are suitable for benchmark exploration since these are multimodal and have multiple local optima. From the table, it was observed that for the functions f_7, and f_8, BFAFA showed competitive performance with BF-PSO. However, the results were outperformed for all other functions which showed that BFAFA was able to reach the global optimum solutions for almost all multimodal functions. In this manner, the proposed algorithm exhibits good exploration capability as well. The balance of exploration and exploitation was tested when a composite function was solved using all algorithms. Here, BFA- FA proved to be the best optimizer for solving the function f_{12}. So, it can be concluded that BFAFA can establish exploration-exploitation balance and give nearly accurate values of global optima.

Table 3.4 shows the single-peak and multi-peak functions. The results of the comparative analysis are shown in Table 3.5. It can be learned from the table that BFAFA gives smaller values for MAE, MAPE, RMSE, and NRMSE as compared to its classical algorithms BFA and FA. This proved the stability of the proposed technique since the error percentage is quite low. The strategy of adaptation and leadership in BFAFA contributes to improving the performance in terms of accuracy since the results of BFAFA outperform BFA.

Another benchmarking model used to test the performance of BFAFA was noisy non-

linear benchmark functions shown in Table 3.6. The evaluation parameter was the processing time (in seconds) for solving the function through the mentioned comparative algorithms. The results of the algorithms when applied over noisy nonlinear functions are shown in Table 3.7. It is quite clear from the results that BFAFA seems to solve noisy functions in less time, hence it is potentially more powerful in solving nonlinear functions. The accuracy index parameters for evaluation were the mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and the normalized root mean square (NRMSE). These parameters are evaluated as follows:
$MAE = \sum_{i=1}^{s} \left| \frac{y_i - f_i}{s} \right|$

$RMSE = \sqrt{\frac{\sum_{i=1}^{s}(y_i - f_i)^2}{s}}$

$NRMSE = \sqrt{\frac{s\sum_{i=1}^{s}(y_i - f_i)^2}{\sum_{i=1}^{s} y_i^2}}$

where s is the total population, y_i is the obtained fitness value, f_i is the actual fitness value for the i^{th} iteration.

Table 3.4: Single peak and multi-peak functions used are as follows

Function name	Function	Range	Fitness Value
Needle-in-haystack	$SP1(x,y) = [(3/(0.05+x^2+y^2)]^2 + [(x^2+y^2)]^2$	$x, y(5,-5)$	3600
Branin	$SP2(x,y) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s$	$x_1[-5,10], x_2[0,15], a = 1,$	
		b=5.1/((4^2)), $c = 5/, r = 6, s = 10, t = 1/8$	0.397887
Extended Beale	$MP3(x,y) =_(i = 1)^(n/2) + [[(1.5 - x_2 i - (1 - x_2 i))]^2 +$		
	[(2.25-x$_($$2i-1)(1-x_2 i^2))]^2]$	$x[-4.5, 4.5]$	$f(x*) = 0, x* = (3, 0.5, \ldots, 3, 0.5)$
Griewank	$MP4(x,y) = 1/4000_(i=1)^n + x_i^2 -_(i = 1)^n + cos\|x_i/i\| + 1$	$x(300, -300)$	$f(x*) = 0, x* = (0, \ldots, 0)$
MCCORMICK	$MP5(x,y) =_(i = 1)^(n-1) + [((-1.5x_i + 2.5x_(i+1) + 1 +$		
	(x$_($$i+1) + x_i)^2) + sin(x_(i+1) + x_i))]$	$fx_0 = [1, 1, ..., 1]$	$f(x*) = -1.9133, x* = (-0.54719, -1.54719, \ldots, -0.54719, -1.54719)$
Drop-Wave	$MP6(x,y) = (1 + cos(12(x_1^2 + x_2^2)))/(0.5(x_1^2 + x_2^2) + 2)$	$x[-5.12, 5.12]$	$f(x*) = -1, x* = (0, \ldots, 0)]$

Table 3.5: Accuracy indexes of BFAFA and other algorithms on single peak and multi-peak functions

Algorithm	Benchmark functions	MAE	MAPE	RMSE	NRMSE
BFA	SP1	52.237	3.9621	71.231	0.0748
FA	SP1	57.328	4.2886	75.965	0.0748
BFAFA	SP1	**50.013**	**3.1385**	**59.893**	**0.0367**
BFA	SP2	62.434	4.3892	65.937	0.0447
FA	SP2	64.329	5.9327	60.332	0.0632
BFAFA	SP2	**57.047**	**4.2037**	**60.023**	**0.0334**
BFA	MP3	80.328	9.3208	90.328	0.0932
FA	MP3	99.327	8.3289	110.094	0.0863
BFAFA	MP3	**71.943**	**6.2647**	**85.392**	**0.0632**
BFA	MP4	89.394	6.3727	98.438	0.0847
FA	MP4	65.048	5.6304	74.731	0.0743
BFAFA	MP4	**60.322**	**4.6533**	**69.329**	**0.0614**
BFA	MP5	75.348	9.4389	79.439	0.0463
FA	MP5	74.243	8.4663	74.482	0.0623
BFAFA	MP5	**74.0127**	**7.0456**	**71.958**	**0.0418**
BFA	MP6	92.473	9.3921	75.923	0.0958
FA	MP6	89.431	8.3947	73.042	0.0963
BFAFA	MP6	**87.839**	**8.2894**	**73.385**	**0.0762**

Table 3.6: Noisy non-linear functions used are as follows

Type of Function	Function
Four Peak Function	$NL1(x,y) = e^((-(x-4)^2-(y-4)^2)) + e^((-(x+4)^2-(y-4)^2)) + 2[e^((-x^2-y^2)) + e^((-x^2-(y+4)^2))]$
Parabolic Function	$NL2(x,y) = 12 - ((x^2+y^2))/100$
Camelback Function	$NL3(x,y) = 10 - log(x^2 - (4 - 2.1x^2 + (1/3)x^4)) + xy + 4y^2(y^2-1)$
Styblinski Function	$NL4(x,y) = 275 - [((x^4 - 16x^2 + 5x)/2) + ((y^4 - 16y^2 + 5y)/2) + 3]$
Goldstein-Price Function	$NL5(x,y) = 10 + log[1/(1 + (1+x+y)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)) * 1/((30 + (2x-3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)))]$
Rosenbrock Function	$NL6(x,y) = 70[(([20 - (1 - x/(-7))^2 + ((y/6) + (x/(-7))^2)^2] + 150])/170] + 10$
Rastrigin Function	$NL7(x,y) = 80 - [20 + x^2 + y^2 - 10(cos(2x) + cos(2y))]$

Table 3.7: Performance of BFAFA and other algorithms when applied on noisy nonlinear benchmark functions in terms of processing time (in sec)

Function	Proposed BFAFA	FA	ABC	BF-PSO	MBO	SSA	BFA
NL1	**51.47**	51.49	51.74	51.68	52.68	51.53	51.87
NL2	**52.73**	52.63	52.81	52.17	52.79	52.74	52.65
NL3	**52.39**	52.39	52.73	52.94	53.04	53.02	53.23
NL4	**53.23**	53.23	56.24	56.34	53.48	54.28	54.29
NL5	**53.11**	53.58	53.82	53.67	53.18	53.12	54.23
NL6	**54.01**	54.85	54.26	54.83	54.94	54.23	54.39
NL7	**51.93**	52.43	52.23	51.98	52.49	51.99	52.95

To test the robustness, effectiveness, and stability of the proposed BFAFA, the algorithms were compared over modern IEEE CEC_2017 Benchmark functions, since it is quite a challenging test suite and involves unimodal, multimodal, composite, and hybrid functions. The description of benchmark functions in detail is mentioned in Table 3.8. The comparison among algorithms is taken over 30 independent runs for all the algorithms. The average and standard deviation are used as performance measures. The evaluation results are shown in Table 3.9. It can be observed that BFAFA showed improved average results in comparison to other algorithms for modern benchmark functions. The average scores by BFAFA were better than other comparative algorithms for the functions C01, C02, C05, C07, C08, C10, C13, C16, C18, C19, C22, and C23. The functions C05, C8, and C10 were solved accurately using BFA- FA, BF-PSO, and SSA. The second rank was achieved by BF-PSO for its good performance over 9 CEC functions, followed by BFA. Also, another interesting observation was that the standard deviation of BFAFA is least for most of the CEC functions. It can be learned that the algorithm is quite stable in solving the problems, besides being stochastic. These observations proved the superiority of the algorithm.

Table 3.8: CEC 2017 optimization functions explained:

Function Type	Function Number	Function Name
Unimodal function	C01	Shifted and Rotated Bent Cigar
	C02	Shifted and Rotated Sum of Different Power
	C03	Shifted and Rotated Zakharov
Simple Multimodal function	C04	Shifted and Rotated Rosenbrock
	C05	Shifted and Rotated Rastrigin
	C06	Shifted and Rotated Expanded Schaffer F6
	C07	Shifted and Rotated Lunacek Bi-Rastrigin
	C08	Shifted and Rotated Non-Continuous Rastrigin
	C09	Shifted and Rotated Levy
	C10	Shifted and Rotated Schwefel
Hybrid function	C11	Zakharov; Rosenbrock; Rastrigin
	C12	High-conditioned Elliptic; Modified Schwefel; Bent Cigar
	C13	Bent Cigar; Rosenbrock; Lunacek bi-Rastrigin
	C14	High-conditioned Elliptic; Ackley; Schaffer F7; Rastrigin
	C15	Bent Cigar; HGBat; Rastrigin; Rosenbrock
	C16	Expanded Schaffer F6; HGBat; Rosenbrock; Modified Schwefel
	C17	Katsuura; Ackley; Expanded Griewank plus Rosenbrock; Schwefel; Rastrigin
	C18	High-conditioned Elliptic; Ackley; Rastrigin; HGBat; Discus
	C19	Bent Cigar; Rastrigin; Griewank plus Rosenbrock; Weierstrass; Expanded Schaffer F6
	C20	HappyCat; Katsuura; Ackley; Rastrigin; Modified Schwefel; Schaffer F7
Composite functions	C21	Rosenbrock; High-conditioned Elliptic; Rastrigin
	C22	Rastrigin; Griewank; Modified Schwefel
	C23	Rosenbrock; Ackley; Modified Schwefel; Rastrigin
	C24	Ackley; High-conditioned Elliptic; Griewank; Rastrigin
	C25	Rastrigin; HappyCat; Ackley; Discus; Rosenbrock
	C26	Expanded Schaffer F6; Modified Schwefel; Griewank; Rosenbrock; Rastrigin
	C27	HGBat; Rastrigin; Modified Schwefel; Bent Cigar; High-conditioned Elliptic; Expanded Schaffer F6
	C28	8 Ackley; Griewank; Discus; Rosenbrock; HappyCat; Expanded Schaffer F6

Table 3.9: Fitness function f(.) evaluation on CEC_2017 benchmark functions over 30 independent runs

Function	Proposed BFAFA		BFA		FA		ABC		BF-PSO		MBO		SSA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
C01	0.00e+00	0.00e+00	6.03e-01	1.00e+00	1.08e-01	5.47e-01	6.05e-01	1.00e+00	0.00e+00	0.00e+00	3.52e-01	5.69e-02	5.82e-03	2.84e-03
C02	0.00e+00	0.00e+00	3.36e-01	2.15e-01	2.73e-01	3.68e-02	3.50e-01	1.59e-01	0.15e+00	0.03e+00	3.48e-01	1.90e-01	0.01e-01	0.83e-02
C03	3.60e+04	5.13e+04	3.00e+05	2.41e+04	8.73e+03	6.90e+03	3.52e+04	3.00e+04	1.23e+03	4.32e+02	5.35e+03	6.83e+03	5.34e+04	7.23e+03
C04	1.30e+01	5.40e-02	4.59e-02	6.30e-03	2.67e+01	9.57e+00	1.57e+01	6.74e-01	4.77e+01	1.23e-01	4.66e+01	5.67e-03	1.94e+01	1.20e-01
C05	0.00e+00	0.00e+00	1.52e-01	2.00e-03	0.00e+00	0.00e+00	6.79e-01	8.30e-01	0.00e+00	1.10e-01	4.56e-01	1.40e-01	0.00e+00	4.00e-03
C06	6.02e+02	1.08e+02	7.01e+02	1.01e+02	1.10e+03	2.89e+02	7.59e+03	4.35e+03	4.23e+02	7.90e+03	4.56e+03	5.18e+02	8.03e+02	5.03e+02
C07	4.25e+01	2.64e+01	5.32e+01	2.91e+01	5.63e+01	3.84e+01	6.13e+03	1.50e+01	6.40e+01	2.37e+01	5.43e+01	2.77e+01	6.78e+01	3.42e+01
C08	0.00e+00	0.00e+00	0.72e+00	0.02e-01	0.03e+00	0.14e-02	7.24e-02	7.50e-03	0.00e+00	0.00e+00	3.48e-02	4.50e-03	0.00e+00	5.60e-04
C09	4.00e-01	2.40e-03	3.20e-01	3.12e-01	7.34e-01	5.60e-03	8.30e-02	7.56e-03	2.43e-01	5.60e-03	7.83e-01	5.20e-02	6.40e-01	2.34e-04
C10	0.00e+00	8.80e-16	0.16e-01	1.03e-02	2.67e-01	7.82e-07	4.58e-02	3.14e-02	0.00e+00	9.50e-14	3.40e-13	9.91e-11	0.00e+00	9.24e-03
C11	2.30e-02	2.57e-17	1.32e-01	3.00e-07	5.67e-02	7.34e-01	8.36e-01	2.71e-01	6.70e-03	4.57e-12	5.67e-01	5.60e-01	3.10e-01	2.90e-10
C12	5.90e-12	2.53e-10	6.60e-12	4.63e+01	7.16e-01	0.00e+00	5.80e-02	7.87e-01	3.62e-01	7.56e-11	6.46e-21	6.50e-09	6.00e-10	5.00e-12
C13	0.00e+00	0.00e+00	3.20e-01	2.30e+00	1.17e-01	6.75e-02	4.16e-02	1.58e-01	1.20e-01	0.00e+00	5.60e-01	6.70e-01	0.00e+00	5.00e-03
C14	2.00e-20	5.60e-22	4.30e-30	2.30e-22	7.35e-02	4.76e-02	4.23e-14	1.26e-20	1.23e-03	7.82e-15	6.00e-12	4.65e-05	4.20e-18	4.00e-20
C15	7.12e+00	1.34e+00	1.92e+01	1.23e+01	1.87e+01	1.08e+01	8.12e+00	6.35e+00	4.12e+00	9.23e+00	8.34e+00	7.98e-01	8.34e+00	2.77e+00
C16	3.05e+00	8.09e-08	1.60e+01	5.00e-05	2.57e+01	5.68e-01	5.64e+00	3.60e+00	5.12e+00	3.23e-05	1.25e+01	9.89e+00	5.06e+00	9.03e-07
C17	9.67e-01	6.34e-20	1.18e+00	3.24e+00	5.65e-01	4.59e-01	8.53e-01	4.63e-03	4.50e-01	3.54e-17	4.70e-01	7.20e-10	1.40e+00	5.23e-18
C18	2.61e+03	1.63e+03	2.67e+05	5.27e+04	5.84e+07	8.75e+04	5.48e+05	5.69e+03	2.32e+04	1.42e+01	4.66e+04	1.23e+03	3.21e+03	2.57e+03
C19	0.00e+00	0.00e+00	0.36e-05	0.84e-06	4.37e-01	4.50e-01	0.00e+00	0.00e+00	0.00e+00	8.30e-01	4.49e-03	1.20e-04	0.91e+00	0.31e-01
C20	9.50e-02	5.40e-03	3.23e-03	7.33e-01	6.78e-01	6.79e-02	1.67e-01	2.55e-03	3.19e-01	6.23e-02	4.70e-03	7.50e-03	1.62e-01	6.70e-03
C21	3.90e+00	0.00e+00	1.25e-01	6.33e-01	2.86e+01	4.54e+00	1.02e+01	1.90e-01	2.90e+00	3.20e-01	8.56e+00	4.50e-03	4.70e+00	0.00e+00
C22	3.17e-01	4.56e-01	5.27e-01	7.29e-01	5.87e-01	4.37e-01	4.25e-01	2.30e-01	6.72e-01	9.45e-01	9.46e-01	6.50e-01	6.53e-01	3.79e-01
C23	4.32e-01	2.53e-01	5.74e+00	2.64e-01	6.75e+01	4.56e+01	6.75e+00	1.20e-03	7.43e+00	6.78e-01	1.56e+01	1.20e-01	6.73e+00	5.60e-01
C24	3.46e+01	2.34e-15	0.00e+00	5.27e-01	6.86e+01	4.55e+01	4.28e+01	1.25e-01	6.76e+01	8.56e-15	6.77e+01	3.40e-13	4.62e+01	5.60e-13
C25	2.92e+01	7.34e-01	7.98e+02	4.56e+00	7.46e+01	5.06e+01	6.12e+00	1.12e+00	1.24e+01	1.48e+01	8.47e+01	1.20e-03	3.54e+01	8.95e-01
C26	4.70e-01	3.20e-01	6.50e-01	6.40e-01	5.68e-01	3.64e-01	1.65e-01	4.32e-01	7.46e-01	6.45e-01	5.67e-01	6.00e-01	5.80e-01	7.85e-01
C27	2.13e+03	1.22e+03	2.35e+05	4.21e+03	5.90e+04	6.87e+03	3.24e+05	4.23e+02	6.22e+05	1.23e+03	4.59e+00	5.33e+03	3.12e+03	2.48e+03
C28	1.39e+01	8.37e+00	4.23e+02	6.70e+01	1.68e+01	6.67e+00	1.54e+01	6.87e-01	2.30e+01	9.15e+00	2.79e+00	9.12e+00	1.48e+01	9.43e+00

The third experiment was comparing the time complexities of the proposed approach concerning the standard classical counterparts, i.e., BFA and FA. Table 3.10 shows the time complexity of BFAFA, BFA, and FA. The convergence rate tells the time taken by the algorithm to reach the global optima. From the table, it was concluded that BFAFA converges faster as compared to BFA and FA in most cases except the functions f_2, f_7, f_9, and f_{10}, where FA performed better than BFAFA. Another observation was that for the composite function f_{12}, BFAFA outperformed BFA and FA. It is important to note the performance of such functions since most of the algorithms fail or face difficulty in optimizing the composite functions on time. Hence, BFAFA can be used to give optimal results for composite functions in less time. It is interesting to note that generally after the hybridization process, the time taken generally increases due to additional function evaluations, however, the time taken by BFAFA is less or comparable. This is because the elimination and dispersion steps were removed in BFAFA. This step in classical BFA involved the complexity of eliminating some parts of the swarm of bacteria and dispersing some other parts of the swarm to a random location. But in BFAFA, when BFA is hybridized with FA, which has a randomness component, the overall function evaluations are less compared to classical BFA. The fourth experiment is performed by testing the scalability in the multi-dimensional environment. Sometimes, it is observed that the algorithms perform well in lower dimensions, but give an absurd performance in higher dimensions. To test the performance of BFAFA over higher and lower dimensions, a comparison of BFAFA is done with BFA and FA over 20, 30, 50, and 100 dimensions in Table 3.11. The results demonstrate that BFAFA outperforms the other two algorithms and offers feasible results for 20, 30, and 50 dimensions. For 100-D, the results are slightly improved giving scope for more improvement in the future. However, they are still better than the traditional BFA and FA for the optimization functions taken.

Table 3.10: Convergence time of BFAFA compared to traditional BFA and FA

Function	Time (sec)		
	BFA	FA	Proposed BFAFA
f1	58.44	56.58	56.36
f2	57.49	57.45	57.78
f3	58.69	56.97	56.32
f4	57.82	56.85	56.21
f5	57.58	58.75	55.63
f6	58.68	57.45	56.67
f7	56.69	55.22	57.40
f8	56.25	57.08	53.44
f9	59.04	57.42	57.83
f10	59.62	56.22	56.49
f11	58.41	52.36	52.12
f12	55.87	59.53	55.40

Table 3.11: Experimental results with higher dimensions and their comparison with conventional algorithm

Optimization function	Dimensions	Optimum Value, f(.)		
		BFA	FA	BFAFA
f1	20-D	9.887e-01	1.4833e-04	7.1293e-16
	30-D	0.1582e-01	2.9127e-03	8.8004e-06
	50-D	0.9923e-02	3.3939e-01	1.7175e-03
	100-D	2.4123e-01	2.0834e-01	5.7119e-02
f5	20-D	0.3478e-01	1.8241e-01	0.01604e-04
	30-D	0.8432e+00	2.9127e+00	0.1184e+00
	50-D	0.5461e+00	2.2342e+00	0.1427e+00
	100-D	0.6923e+00	0.8982e+00	0.3133e+00
f8	20-D	5.237e-05	1.3421e-07	6.3287e-18
	30-D	0.5933e-04	1.1284e-03	0.4438e-10
	50-D	0.2321e-01	0.8756e02	0.455e-05
	100-D	7.8673e+00	6.9223e+00	5.4312e+00
f10	20-D	9.3735e-11	2.4833e-13	5.133e-30
	30-D	0.01258e-08	0.8762e-10	0.3622e-18
	50-D	0.3552e-05	2.2398e-06	1.1732e-11
	100-D	2.7724e-01	2.8923e-01	1.7193e-02
f12	20-D	3.03e+00	3.9230e+00	3.0000e+00
	30-D	3.9730e+00	4.1730e+00	3.7600e+00
	50-D	4.1277e+00	4.2873e+00	4.1700e+00
	100-D	4.8123e+00	4.7662e+00	4.6124e+00

The fifth experiment is performed by testing the significance of the results (Wilcoxon rank-sum test at 5% significance level and friedman rank test) (116)(117). Wilcoxon test is a non-parametric statistical test that verifies whether or not both sets of solutions are significant. It returns a p-value that defines the degree of significance. If the p-value is less than 0.05, an algorithm is statistically significant. This is done by choosing the best algorithm in each run and comparing it with other algorithms independently. Also, since the best algorithm cannot be compared to itself, hence N/A (Not Applicable) is written. Since the metaheuristic algorithms are stochastic, therefore it is required to test the significance. The wilcoxon test results are shown in 3.12. It can be observed from the table that the BFAFA algorithm was able to perform well for the test functions C01, C02, C04, C06, C07, C09, C11, C14, C15, C17, C20, C21, C22, C24, C25, and C28. After BFAFA, the algorithms BF-PSO and SSA were ranked second since they gave good performance on three test functions each. From the statistical results, it can be said that BF-PSO and SSA were able to give good results and outperform other algorithms, however, BFAFA gave outstanding performance and proved superior. Friedman test is also a non-parametric test, which is based on the average ranked value. If the p-value level of significance) is less than or equal to 0.05, then the null hypothesis is rejected. This means that the statement that all the algorithms perform equally well, is not true. This verifies whether the proposed algorithm performs statistically significant than other algorithms or not. The average rankings of all the algorithms used in this study on the test suites unimodal, multi-modal, composite, and CEC_2017 functions using the friedman rank test are shown in Table 3.13. The statistical value of the test came out to be 108.10, and the p-value of BFAFA was 4.923E-06 which proved that there was a significant difference between BFAFA and other algorithms. BFAFA was followed by BF-PSO, SSA, MBO, BFA, FA, and ABC. The results show the superior and significant performance of BFAFA over others for most of the functions. Hence, the proposed approach can solve problems efficiently and stands at the first rank for this benchmark suite.

By observing the discussions and findings, it is evident that BFAFA edged over most algorithms taken in this study in terms of efficiency, significance, and attaining global optima results. The convergence curve showed a smooth transition from exploration to exploitation within the search space. As compared to its classical part-and-parcels, the proposed approach BFAFA possesses superior convergence capability for all optimization problems. The results for multimodal functions are superior when solved by BFAFA, hence local optima avoidance is ensured to a much larger extent. The time taken by PSO-BOA is comparable and not more than its components, i.e., BFA and FA. The proposed algorithm is significantly better and stands on a better rank than other algorithms taken in this study.

Table 3.12: Statistical Analysis using P-values obtained from the wilcoxon rank-sum test

CEC Functions	BFA	FA	ABC	BF-PSO	MBO	SSA	Proposed BFAFA
C01	4.380e-04	1.089e-03	7.260e-02	5.4002e-01	7.182e-03	3.251e-04	N/A
C02	1.145e-03	4.513e-03	1.0528e-04	3.14292e-01	1.674e-06	6.180e-04	N/A
C03	3.022e-02	4.506e-03	1.748e-11	N/A	9.668e-02	4.603e-03	4.502e-03
C04	1.592e-03	3.886e-03	1.083e-04	2.6104e-04	2.2121e-10	6.219e-02	N/A
C05	8.126e-03	1.091e-03	5.138e-01	N/A	1.868e-03	5.4725e-02	1.093e-03
C06	2.566e-02	4.084e-02	7.410e-05	7.837e-05	9.437e-02	5.218e-15	N/A
C07	4.001e-01	3.0614e-01	0	7.13e-03	3.424e-02	7.85e-03	N/A
C08	1.110e-16	N/A	2.887e-15	0	4.273e-02	4.60e-03	7.74e-03
C09	1.568e-02	4.281e-02	0	8.089e-02	0	0	N/A
C10	9.83e-02	N/A	2.162e-04	7.332e-02	9.915e-02	7.832e-02	4.325e-02
C11	3.234e-02	0	8.862e-02	1.996e-08	0	6.6e-04	N/A
C12	1.243e-02	6.31e-03	0	8.184e-02	N/A	9.977e-02	9.960e-02
C13	6.6502e-01	3.3162e-05	4.160e-02	0	1.65e-03	N/A	6.745e-02
C14	3.214e-02	0	1.197e-04	0	0	6.523e-02	N/A
C15	3.4e-04	6.320e-02	3.226e-02	2.36e-03	6.3e-04	2.260e-02	N/A
C16	4.672e-02	6.4e-03	3.54e-02	0	7.3e-04	N/A	6.320e-02
C17	1.372e-02	8.376e-01	1.026e-02	0	0	0	N/A
C18	1.026e-02	5.412e-02	5.0685e-12	N/A	9.732e-02	5.408e-02	1.026e-02
C19	N/A	8.672e-02	9.069e-02	8.4095e-01	6.6e-04	9.977e-02	9.960e-02
C20	1.231e-02	6.968e-02	4.441e-16	0	0	0	N/A
C21	2.172e-02	8.278e-13	6.472e-13	0	1.716e-08	0	N/A
C22	3.423e-02	4.500e-02	1.903e-02	5.012e-01	2.9089e-01	2.21e-03	N/A
C23	N/A	0	1.981e-04	0	1.112e-16	0	5.001e-02
C24	3.428e-02	0	2.0096e-02	0	0	0	N/A
C25	2.133e-02	0	1.009e-02	0	2.01e-03	2.276e-05	N/A
C26	5.323e-02	4.367e-02	5.306e-01	8.96e-03	2.411e-02	N/A	7.400e-02
C27	N/A	1.112e-15	0	0	2.348e-02	9.985e-06	2.722e-02
C28	4.382e-02	2.018e-13	4.008e-01	5.792e-02	3.023e-02	1.009e-02	N/A

Table 3.13: Average rankings of algorithms using friedman's test

Algorithm	standard functions f1-f12	CEC_2017 functions
BFAFA	**1.894735**	**2.30**
BFA	5.284942	3.63
FA	5.928479	4.29
ABC	6.023848	3.49
BF-PSO	2.024725	4.20
MBO	3.928472	5.80
SSA	3.283682	4.78

3.5 Application of the BFAFA algorithm to structural design problems

This section applies the proposed BFAFA algorithm to solve two structural design problems: The Cantilever beam design problem and the Three-bar truss design problem. The below section will show the best-obtained design and the function evaluations required to achieve it, during 10 runs.

3.5.1 Cantilever beam design problem

A Cantilever beam (118) consists of five elements that are hollow and have a square-shaped cross-section. As shown in Fig. 3.3, there are 5 structural parameters, each element being defined by a variable. Node 1 is rigidly supported and a vertical load is applied on Node 6. The optimization problem is formulated as follows:

Consider $\vec{x} = [x_1 x_2 x_3 x_4 x_5]$

Minimize $f(\vec{x}) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5)$

Subject to $g(\vec{x}) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0,$

Variable range $0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100.$

Table 3.14 shows the solution to the problem using the proposed approach and its comparison with other techniques like Symbiotic Organisms Search (SOS) (119), CS (120), Method of Moving Asymptotes (MMA) (121), Generalized Convex Approximation (GCA) first and second derivatives- GCA I (122), GCA II (122), ABC (113), MBO (114), SSA (4). The optimum weights given by the algorithms SOS, ABC, MBO, and SSA were similar, however, BFAFA gave lesser value, which give an edge to its performance. Similarly, it can be seen that the functions evaluations required to reach the global optimum were optimum in the case of BFAFA, followed by SSA, MBO, ABC, and SOS. CS gave the output in least function evaluations i.e., 2500, but the optimal weight was higher. So, BFAFA proves to be a good optimizer that delivers optimal value in reasonable function evaluations.

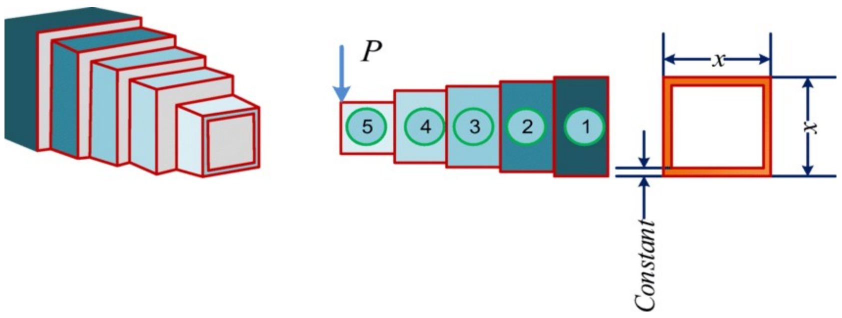


Figure 3.3: Cantilever beam design problem

Table 3.14: Results for cantilever design problem using proposed approach and other algorithms

Algorithm	Optimal value for variables					Optimum weight	Max. function evaluations
	x_1	x_2	x_3	x_4	x_5		
BFAFA	6.014752	5.30716	4.48352	3.49965	2.15327	1.33995	12000
SOS	6.01878	5.30344	4.49587	3.49896	2.15564	1.33996	15000
CS	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999	2500
MMA	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400	NA
GCA I	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400	NA
GCA II	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400	NA
ABC	6.01812	5.31142	4.48836	3.49751	2.158329	1.33996	14000

3.5.2 Three-bar truss design problem

The three-bar truss design problem (118) is a highly constrained problem where one has to minimize its weight. The problem is formulated as follows: Consider $\vec{x} = [x_1 x_2] = [A_1 A_2]$,

Minimize $f(\vec{x}) = (2\sqrt{2}x_1 + x_2) * l$

Subject to $g_1(\vec{x}) = \frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P - \sigma \leq 0,$

$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1+2x_1x_2}P - \sigma \leq 0,$

$g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2+x_1}P - \sigma \leq 0,$

Variable range $0 \leq x_1, x_2 \leq 1,$

Where $l = 100cm, P = 2(KN)/(cm^2), \sigma = 2(2KN)/(cm^2)$

The three-bar truss is shown in Fig. 3.4, and the solution is shown in Table 3.15, which shows the solution using the proposed approach and other techniques. The other techniques used are Differential evolution with dynamic stochastic selection (DEDS) (123), BF-PSO (115), Mine blast algorithm (MBA) (124), Ray and Sain (125), Tsa (126), CS (120), ABC (113). MBO (114), and SSA (4). It is clear from the results that BFAFA stood first in optimizing the three-bar truss problem with minimum weight. The function evaluations were also reasonably less. Between ABC and BFAFA, although the function evaluations were lower for ABC, the global optimum value was better delivered by BFAFA. CS gave the optimal value in the least function evaluations but the value was a little higher. Overall, a good deal would be to get the minimum optimization value in reasonable function evaluations. Although BFAFA was competitive with other state-of-art techniques, its overall performance was better.

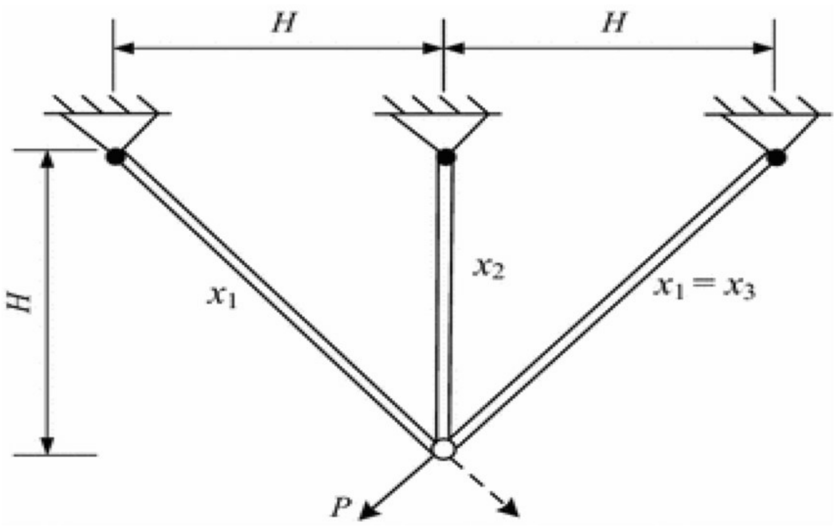


Figure 3.4: Three-bar truss design problem

Table 3.15: Results for three-bar truss design problem using proposed approach and other algorithms

Algorithm	Optimal value for variables		Optimum Weight	Max. Function evaluations
	x_1	x_2		
BFAFA	0.7886715	0.4084389	263.89582382	12000
DEDS	0.78867513	0.40824828	263.8958434	15000
BF-PSO	0.7886751	0.4082482	263.8958433	17600
MBA	0.7885650	0..4085597	263.8958522	20000
Ray and Sain	0.795	0.395	264.3	NA
Tsa	0.788	0.408	263.68	NA
CS	0.78867	0.40902	263.9716	15000
ABC	0.7886628160 0 0317	0.408283133832901	263.8958434	11800
MBO	0.788897555578973	0.407619570115153	263.895881496069	13000
SSA	0.788665414258065	0.408275784 4 4 4547	263.8958434	12500

3.6 Conclusion

This chapter proposes two novel improvements in the conventional BFA for solving multi-modal optimization problems and convex landscapes with fast convergence, scalability, and efficiency, in less time. Also, the main goal is to achieve the global optimum value while balancing exploration as well as exploitation. To achieve this, firstly, a self-adaptive strategy is proposed where the step size of the bacteria is dynamically changed using the success percentage of bacteria and examining their state after each iteration, to reach the global optima. It leads to better exploration-exploitation balance, improved convergence time, accuracy close to the global optimal value, and good results for multimodal optimization problems. Secondly, in place of the elimination and dispersion step, the leadership strategy is applied where the bacterium with the least fitness value (most healthy) is the leader and the other bacteria in the swarm follow the leader using the firefly algorithm position equation. It leads to lesser time complexity, fewer computations, and fast convergence since FA is known for its ability to converge faster.

Also, therefore there is no addition of newer calculations, so it leads to decreased time complexity.
To benchmark the performance of the proposed algorithm, a series of tests were conducted. The convergence curves were studied and compared with other algorithms. The proposed algorithm was tested on 12 standard, 7 nonlinear, 2 single peaks, and 4 multi peaks, as well as 28 CEC_2017 benchmark functions, and compared to the classic as well as recent nature-inspired algorithms to validate its performance. For this several evaluation parameters were used like optimal value, mean, standard deviation, processing time, MAE, MAPE, RMSE, and NRMSE. The performance was tested on a multi-dimensional landscape as well to test its scalability. The significance of the results was determined under Wilcoxon Statistical as well as Friedman rank test analysis at a 5% significance level. The overall result was that BFAFA performed better than other well-known algorithms in the literature, were able to converge quickly, exhibited stability, gave good results on multi-modal functions, was scalable in multi-dimensional cases, and was statistically significant. BFAFA was applied to two classical engineering problems (a Cantilever design problem and a three-bar truss design problem) and the results were compared to other algorithms that are efficient in solving the problem. The results proved that BFAFA can solve real-world problems also with lesser function evaluations.
The main contributions of the proposed BFAFA are as follows:

i To increase the convergence and establish exploration-exploitation balance, a novel hybrid of BFA and FA is proposed, where adaptive strategy and leadership strategy are implemented.

ii The proposed method has been extensively tested on benchmark functions (standard, single peak, multi-peak, noisy non-linear, and CEC_2017). The results outperform the other competitive algorithms in terms of convergence speed, time, mean absolute error, mean square error, mean, standard deviation, and other related parameters. It works well in multi-dimensional and multi-modal environments.

iii The algorithm has been successfully implemented on real-world engineering problems, and the results are better than other optimization algorithms.

To summarize, BFAFA finds a better solution without getting trapped in local optima. Due to its stochastic nature, they can search a complicated and uncertain area as well. FA improves the search strategy of BFA and makes it more reliable to provide global optima within reasonable computation time and with better exploration. The performance of the proposed approach has proved to be better than some of the well-established and recent metaheuristic algorithms and can be generalized for other metaheuristics in the literature.

The proposed BFAFA is proven to produce the best results among various large-scale real-world problems. But, in some cases, the improvement is not shown or does not produce higher accuracy. It is due to a lot of parameters, lack of constraints, and lack of adaptation, which will be solved by our next proposed model, explained in our next chapter.

CHAPTER 4

HYBRID PARTICLE SWARM OPTIMIZATION-BUTTERFLY OPTIMIZATION ALGORITHM (PSOBOA)

Most real-world problems have a lot of constraints that need to be satisfied. Also, when using EA to solve constrained-based problems, parameter tuning plays an important role. In this regard, Self-adaptation of parameters plays a key role in giving good results.

4.1 Introduction

Nature-inspired algorithms have a tremendous ability to solve complex real-world problems in reasonable time and cost. Due to this, a lot of researchers have given much attention to this area. Till now, several such algorithms have been used to solve global optimization or real-world combinatorial problems. Some of them are ABC, GA, BFA, Grasshopper Optimization Algorithm (GOA), SSA (4), etc. These metaheuristic algorithms provide global optima and converge faster. However, whenever the problem is complex with several local optima, the algorithm often gets trapped in the local optimum (127). To solve this problem, many hybrid metaheuristic algorithms are developed. These hybrids are more efficient and robust compared to the individual standard algorithms as they have the advantages of both. Also, it has been observed that most real-world applications are subjected to different types of constraints. Such problems are called COP. Many nature-inspired algorithms, as well as their hybrids, have been used to solve such problems and provide high precision, quick convergence, robustness, and maximum performance. However, the problem of premature convergence, poor exploration, or exploitation remains.
For these reasons, we propose a novel hybrid algorithm based on PSO (2) and BOA (7). The main aim of our work is to provide a solution that is better than other algorithms used in the literature in terms of reaching global optimum, convergence speed, and balancing exploration-exploitation. While PSO has been used in solving several COP, BOA has never been used for such problems. This makes it promising and interesting to see its capabilities for solving COP. Both these algorithms have their strengths and weaknesses. PSO has good exploitation ability and rapid convergence but doesn't

explore search space effectively and suffers from premature convergence. On the other hand, BOA has good exploration ability but converges slowly. So, the main objective of this chapter is to develop a hybrid of PSO and BOA where the advantages of both PSO and BOA can be utilized effectively and the global optimum is reached quickly.
The proposed algorithm aims to provide better convergence results, avoid the local optima trap, and make the working adaptive, free from parameter tuning. It also promises to establish a balance between exploration and exploitation abilities by using the key features of both the algorithms, PSO and BOA. Firstly, the objective function is redefined by adding the parameter-free penalty function for handling the constraints and converting the constrained problem into an unconstrained one. This reduces the overall complexity of the problem and helps to solve it in fewer function evaluations. Secondly, a self-adaptive approach has been adopted in PSO and BOA to ensure a smooth transition from exploration to exploitation and no user-based initialization of key parameters. This makes the algorithm free of parameter tuning. Thirdly, to improve the convergence rate and avoid local optima stagnation, a conditional approach has been used in the local and global search of BOA. The proposed algorithm is verified by addressing the COP like pressure vessel design and welded-beam design problem. The numerical results and the convergence curves show that PSOBOA demonstrates outstanding performance as compared to other state-of-the-art and current algorithms.

4.2 Strategies employed in the PSOBOA algorithm

4.2.1 Constrained optimization parameter free penalty function for handling constraints

Non-linear constrained optimization problem is formulated as follows:
Minimize $f(x)$
Subject to

$$\begin{aligned} h_k(x) &= 0; k = 1, 2, \ldots, p \\ g_j(x) &\leq 0; j = 1, 2, \ldots, q \\ lb_i \leq x_i &\leq ub_i; i = 1, 2, \ldots, n \end{aligned} \tag{4.1}$$

where $x =$represents the decision variables along n-dimensions; f is the objective function, lb_i and ub_i are the lower and upper bounds for the i^{th} variable respectively; p being the number of equality constraints and q is the number of inequality constraints. If any solution 'x' satisfies the constraints h_k or g_j, then g_j is considered to be an active constraint at x. Let Xbe the feasible solution set such that $X = \{x\epsilon R \vee g_j(x) \leq 0; j = 1, 2, \ldots, p + q(M)\}$. So, Eq. (4.1) can be rewritten as
Minimize $f(x)$

Subject to

$$\begin{aligned} g_j\left(x\right) &\leq 0; j = 1, 2, \ldots, M \\ lb_i \leq x_i &\leq ub_i; i = 1, 2, \ldots, n \end{aligned} \tag{4.2}$$

The main aim of the COP is to minimize the objective function and simultaneously satisfy the constraints. According to (128), constraint handling can be done using five methods: penalty functions, repair algorithms, hybrid methods, separation of objective functions and constraints, and special operators. Out of these, penalty functions are most commonly used, however, it has a few drawbacks. There is no guarantee for global optima. Also, a lot of parameters are required to be adjusted which increases the problem complexity and increases the number of evaluations. So, to overcome this, a parameter-free penalty function is used in this work. The modified objective function is defined as given in Eq. (4.3).

$$F(x) = \begin{cases} f(x) & if \;\; x \in X \\ \\ f_w + \sum_{j=1}^{M} g_j(x) & if \;\; x \notin X \end{cases} \tag{4.3}$$

where x is the set of solutions and f_w is the objective function value of the worst feasible solution in the entire population. So, the fitness of the infeasible solution depends on the amount of constraint violation and the population of solutions at hand. On the other hand, the fitness of the feasible solution is fixed and equal to the value of the objective function.
In our algorithm, the objective function is redefined by incorporating the parameter-free penalty function and is determined as given in Eq. (4.3). The idea is to make the constrained problem into an unconstrained one. When COPs are solved using penalty functions, the complexity increases due to the addition of a new set of parameters to the objective function. However, in this work, no additional parameters are added, so this step reduces the complexity of the algorithm, and the number of function evaluations will be less, as compared to the general penalty-function approach taken while solving COPs.

4.2.2 Self-adaptive approach in PSOBOA

To overcome poor exploration in PSO, the self-adaptive approach is used on the controlling parameter, w since the first term of the velocity update equation, i.e., wv_i^t controls the exploration ability of PSO. By trying multiple values of w, we found that values between 0.2 and 0.9 give good results. So, the lower range of w, w_{min} is taken to be

0.2 and the upper range w_{max} to be 0.9. The inertia weight w, therefore, is given by the following Eq. (4.4).

$$w\left(t\right) = w_{max} + \left(w_{min} - w_{max}\right) \times log_{10}\left(rand + \frac{10t}{max_{Iter}}\right) \tag{4.4}$$

where, t and max_{Iter} are the current iteration and the maximum number of iterations respectively, $rand$ is a random number between [0,1]. A similar adaptation for the power exponent a in BOA is conducted, where the lower range of a, $a_{min}$is taken to be 0.1 and the upper range a_{max} to be 0.3. The value of a is updated after each iteration as following Eq. (4.5).

$$a\left(t\right) = a_{max} + \left(a_{min} - a_{max}\right) \times log_{10}\left(rand + \frac{10t}{max_{Iter}}\right) \tag{4.5}$$

The logarithmic decrement of inertia weight and power exponent ensures a smooth transition from exploration in the first few iterations to exploitation in the remaining iterations. Hence, a balance is established. There is no need for the user to tune the parameters at any stage. Hence, it works independently of user interference, when the change of parameter is to be done.

4.2.3 Conditional approach for convergence enhancement

To expedite the convergence phenomenon and avoid the trap of local optima, a conditional approach is used in the local and global search phenomenon of BOA. In this, rather than just moving the butterflies randomly towards the best solution, a check is done to validate if the new position is better than the previous position. If yes, then only the movement will take place, else avoided. The global search phase of BOA will be formulated according to Eq. (4.6).

$$pos_i^{t+1} = pos_i^t + \left(r^2 \times Gbest - pos_i^t\right) \times fr_i \tag{4.6}$$

The local search phase is formulated according to Eq. (4.7)

$$pos_i^{t+1} = pos_i^t + \left(r^2 \times pos_j^t - pos_k^t\right) \times fr_i \tag{4.7}$$

where pos_j^t and pos_k^t are the j-th and k-th search agents chosen randomly from the solution space, respectively.

For conditional check in BOA, the $Pbest$ and $Gbest$ values in BOA are updated if the new solution is better than the previous one, as shown in Eq. (4.8) and Eq. (4.9)

$$Pbest_i \leftarrow pos_i^t if F\left(Pbest_i\right) > F\left(pos_i^t\right) \tag{4.8}$$

$$Gbest \leftarrow pos_i^t if F\left(Gbest\right) > F\left(pos_i^t\right) \tag{4.9}$$

4.3 Development of the PSOBOA algorithm

PSO is simple to implement, flexible, robust algorithm, and good at exploitation, but suffers from local optimum trapping. BOA has good exploration but is slow in convergence. However, to avoid premature convergence, global and local search operators of BOA are used. It leads to minimal addition of computational cost. An attempt is made to use the strengths of both the algorithms and accelerate the convergence along with updating the poor solutions. The algorithm aims at giving better solutions for various COPs by following the above-mentioned strategies.

The algorithm starts with the random initialization of particles in the swarm between the bounds of the variables. The objective function is simplified by using a parameter-free penalty function. A random variable $rand1$ is taken based on which the model of the algorithm is chosen. If $rand1 < 0.5$, BOA is implemented, else PSO is implemented.

In BOA implementation, to avoid the butterfly changing a position that is not better than the previous position, a conditional approach is taken, where a check is made to find whether the new position is better than the previous position. If it is better, then update the results. After each iteration, the power exponent is updated logarithmically. This assists in moving towards the solution and improving the convergence. After each iteration, one gets a better solution. Therefore, this condition improves the quality of the solution.

In PSO implementation, the value of inertia weight w decreases logarithmically after each iteration. This ensures a smooth transition from exploration to exploitation and no parameter-tuning is required by the user. After meeting the stopping condition, the best solution is the global optima value. The proposed algorithm is shown in Algorithm 7.

Algorithm 7 PSOBOA Algorithm

```
1. Initialize the population $x_i$(i=1,2...,n) randomly; Objective function $F(x)$
   from Eq. (4.3); the maximum number of iterations $max_{Iter}$; inertia weight
   $w$; switch probability $p$; power exponent $a$; and sensor modality $c$
2. For each search agent $i$
3. Initialize $pos_i$ randomly
4. Initialize $v_i$ randomly
```

```
5. Evaluate the fitness of the search agents $F(pos_i)$ using Eq. (4.3)
6. Initialize $Pbest$ with a copy of $pos_i$
7. Initialize $Gbest$ with a copy of $pos_i$ with the best fitness
8. End for
9. While $t = 1 : max_{Iter}$
10. For each search agent $i$ in swarm
11. Generate random numbers $rand1$ and $rand2$
12. If $rand1 < 0.5$ then
13. Evaluate the fragrance of butterfly using Eq. (1.16)
14. If $rand2 < p$ then
15. Perform global search using Eq. (1.17) and evaluate the new position
16. Else
17. Perform local search using Eq. (1.18) and evaluate the new position
18. Endif
19. Perform conditional check using Eq. (4.8) and Eq. (4.9) and update the values
    of Pbest and Gbest
20. Update the power exponent using Eq. (4.5)
21. Evaluate the new solution and update
22. Else
23. Update $v_i^t$ and $pos_i^t$ according to the Eq. (1.4) and Eq. (1.5)
24. Evaluate the fitness $F(pos_i^t)$
25. $Pbest_i \leftarrow pos_i^t \; if \; F\,(Pbest_i) > F(pos_i^t)$
26. $Gbest \leftarrow pos_i^t \; if \; F\,(Gbest) > F(pos_i^t)$
27. Update the inertia weight using Eq. (4.4)
28. Evaluate the new solution and update
29. End if
30. End for
31. End While
32. Output the best solution found
```

4.4 Application of the PSOBOA model to structural optimization problems and performance comparison with conventional algorithms

To verify the performance of PSOBOA, it is tested on two constrained optimization problems. These problems are: (i) pressure vessel design (129) and (ii) welded beam design (129). These problems have linear as well as non-linear constraints. For each problem, 30 independent runs are performed and the mean and standard deviation (SD) are taken. Also, the comparison with the other algorithms mentioned in the literature is done to verify the quality of solutions given by the proposed approach. Experiments are performed in Matlab. The values of the parameters used are the same as those used in the literature. However, the values of the parameters of our proposed approach are set as follows.

$c_1 = 2, c_2 = 2, w_{max} = 0.9, w_{min} = 0.2,$

$a_{max} = 0.3, a_{min} = 0.1, p = 0.8, maxIter = 100, n = 20$

4.4.1 Pressure vessel design problem

Pressure Design is a classical problem in which the cost related to three sections of a cylindrical pressure vessel, namely formation, material, and welding, must be minimized. Four variables are shell thickness Ts, head thickness T_h, inner radius R, and length of the vessel L. The mathematical equation, along with the constraints, is mentioned as follows:

Consider $\overrightarrow{x} = (x_1 x_2 x_3 x_4] = (T_s T_h R L]$,

Minimize $f\left(\overrightarrow{x}\right) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$,

Subject to

$$\begin{aligned} g_1\left(\overrightarrow{x}\right) &= -x_1 + 0.0193x_3 \leq 0, \\ g_2\left(\overrightarrow{x}\right) &= -x_2 + 0.00954x_3 \leq 0, \\ g_3\left(\overrightarrow{x}\right) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0, \\ g_4\left(\overrightarrow{x}\right) &= x_4 - 240 \leq 0 \end{aligned} \tag{4.10}$$

Variable range

$$\begin{aligned} 0 \leq x_1 &\leq 100, \\ 0 \leq x_2 &\leq 100, \\ 10 \leq x_3 &\leq 200, \\ 10 \leq x_4 &\leq 200 \end{aligned}$$

The statistical results of the proposed algorithm PSOBOA and its comparison with

ABC (130), GGA (131), PSO (2), UDE (132), BOA (7), and GWO (5) are shown in Table 4.1 for 30 independent runs. The friedman rank test (133) is performed on the mean values to test the significance of PSOBOA over other algorithms. The convergence rate was also noted to see the time taken by the algorithms to reach the global optima. It can be verified that the results of optimization using PSOBOA are better than other algorithms in terms of their mean values. The lower value of SD indicates consistent results over different runs, which makes the algorithm more robust. PSOBOA gives a substantially lower value of SD compared to other algorithms, hence it can be concluded that it is a more robust algorithm. This approach outperforms other well-known classical algorithms in solving COP. The mean and SD of PSO and BOA are generally higher than other algorithms. However, when their hybrid is developed, it gives significant results. This is because the hybrid improvises the results by overcoming the drawbacks of its classical algorithm components. This was a very interesting observation from the results. From the Friedman rank test, the order of all the algorithms based on their rank is PSOBOA $>$ ABC $>$ BOA $>$ GGA $>$ GWO $>$ UDE $>$ PSO. From the results, we can see that the rank of PSOBOA is better than other algorithms, especially its counterparts. This indicates that the proposed work is statistically significant in comparison to other works. The convergence rates are in the order ABC $>$ GGA $>$ PSOBOA $>$ PSO $>$ UDE $>$ BOA $>$ GWO. This shows that due to the hybridization process and other significant improvements, the time taken is a little higher in the case of PSOBOA. So, there is a scope for improving the time taken by reducing the complexity of the algorithm.
In addition, the convergence curves of all these algorithms are shown in Fig. 4.1. It is observed in the convergence curve graph that the proposed algorithm converges very quickly than other recent algorithms towards the global optimum value, however, the graphs for individual components PSO and BOA converge at a slower rate for the same problem. Statistical results of BOA are comparative with the proposed approach, but convergence curve analysis shows that due to the addition of local and global search features of BOA in PSO, the convergence phenomenon is much enhanced.

4.4.2 Welded beam design problem

Welded beam design is another minimization problem with four variables namely, the thickness of the weld h, the length of bar$l$, the height of bar t, thickness of bar b. The mathematical equation along with the constraints is mentioned in the following equation.

Consider $\overrightarrow{x} = (x_1 x_2 x_3 x_4] = (hltb]$,

Minimize $f\left(\overrightarrow{x}\right) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 \left(14.0 + x_2\right)$,

Table 4.1: Statistical results of the pressure vessel design problem

Method	Design Variables				Mean	SD	Rank	Time
	x1	x2	x3	x4				
ABC	0.8821	0.4372	42.4298	168.3288	6286.2369	6.8124	1.27	3.23
GGA	0.8125	0.4375	42.0984	176.6365	6329.8352	20.8361	3.63	4.25
PSO	0.8363	0.4837	43.2184	175.2749	6136.2321	102.3312	5.29	5.39
UDE	0.8125	0.4375	42.0984	176.6365	6328.1267	46.1281	4.85	6.98
BOA	0.7832	0.3921	41.3128	170.6217	6672.3271	150.31766	2.19	7.21
GWO	0.8125	0.4375	42.0984	176.6365	6441.9832	83.3126	3.67	10.32
PSOBOA	0.7781	0.3852	40.8743	198.2316	5913.4328	3.8323	1.18	4.81

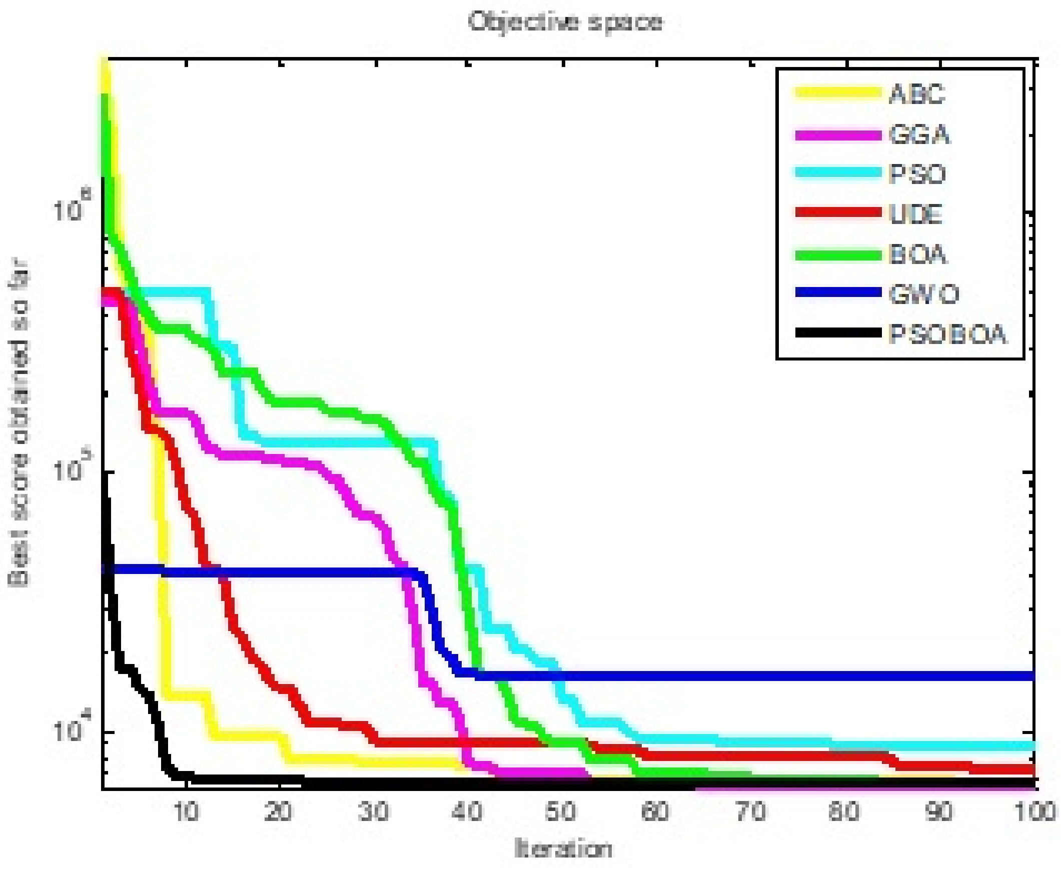


Figure 4.1: Convergence curves of different algorithms for pressure vessel design problem

Subject to

$$\begin{aligned} g_1\left(\overrightarrow{x}\right) &= \tau\left(x\right) - \tau_{max} \leq 0, \\ g_2\left(\overrightarrow{x}\right) &= \sigma\left(x\right) - \sigma_{max} \leq 0, \\ g_3\left(\overrightarrow{x}\right) &= \delta\left(x\right) - \delta_{max} \leq 0, \\ g_4\left(\overrightarrow{x}\right) &= x_1 - x_4 \leq 0, \\ g_5\left(\overrightarrow{x}\right) &= P - P_c\left(\overrightarrow{x}\right) \leq 0, \\ g_6\left(\overrightarrow{x}\right) &= 0.125 - x_1 \leq 0, \\ g_7\left(\overrightarrow{x}\right) = 0.10471x_1^2 + 0.04811x_3x_4\left(14.0 + x_2\right) - 5.0 &\leq 0 \end{aligned} \tag{4.11}$$

Variable range

$$\begin{aligned} 0.1 \leq x_1 \leq 2, \\ 0.1 \leq x_2 \leq 10, \\ 0.1 \leq x_3 \leq 10, \\ 0.1 \leq x_4 \leq 2, \end{aligned}$$

Table 4.2: Statistical results of the welded beam design problem

Method	Design Variables				Mean	SD	Rank	Time
	x_1	x_2	x_3	x_4				
ABC	0.2057	3.4701	9.0417	0.2103	1.7225	1.73E-02	5.32	3.84
GGA	0.2057	3.4703	9.2719	0.2057	1.7294	9.72E-02	4.74	7.99
PSO	0.2059	3.4327	9.3128	0.2046	1.9372	8.73E-01	2.38	6.34
UDE	0.2382	3.4684	9.0472	0.2076	1.7391	7.37E-05	3.55	5.75
BOA	0.2058	3.3294	9.4723	0.2103	1.7294	8.24E-04	2.19	3.12
GWO	0.2059	3.7324	9.3826	0.2013	1.7063	2.18E-03	1.90	3.52
PSOBOA	0.2057	3.2612	9.0321	0.2011	1.6992	1.83E-06	1.26	3.84

$$\tau\left(\overrightarrow{x}\right) = \sqrt{\tau'^2 + 2\tau'\tau''\frac{x_2}{2R} + \tau''^2}$$
$$\tau' = \frac{P}{\sqrt{2}x_1x_2}$$
$$\tau'' = \frac{MR}{J}$$
$$M = P\left(L + \frac{x_2}{2}\right)$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$$
$$J = 2\Big(\sqrt{2}x_1x_2\Big(\frac{x_2^2}{4}\big(\frac{x_1+x_3}{2}\big)^2\Big]\Big]\Big\}$$
$$\sigma\left(\overrightarrow{x}\right) = \frac{6PL}{x_4x_3^2}$$
$$\delta\left(\overrightarrow{x}\right) = \frac{4PL^3}{Ex_3^2x+x_4}$$
$$P_c\left(\overrightarrow{x}\right) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$
$$P = 6000lb, L = 14 \in ., \delta_{max} = 0.25 \in ., E = 30X10^6psi,$$
$$G = 12X10^6psi, \tau_{max} = 13600psi, \sigma_{max} = 30000psi$$

The statistical results of the proposed algorithm PSOBOA and its comparison with ABC (130), GGA (131), PSO (2), UDE (132), BOA (7), and GWO (5) are shown in Table 4.2 for 30 independent runs. The rank of algorithms is depicted to test the significance, and the time taken by the algorithm is also noted. It can be observed that PSOBOA gives the best global optima results compared to other algorithms. PSOBOA has better mean values and SD is lowest as compared to other works. The rank of all the algorithms is in the order PSOBOA $>$ GWO $>$ BOA $>$ PSO $>$ UDE $>$ GGA $>$ ABC and the convergence time is in the order BOA $>$ GWO $>$ PSOBOA = ABC $>$ UDE $>$ PSO $>$ GGA. The results show that PSOBOA gives significant results in comparison to the other algorithms mentioned in the literature. The convergence time is compromised to some extent; however, the algorithm is statistically significant. It is interesting to note that GWO gives equally competitive results for the Welded beam design problem, unlike the Pressure vessel design problem, where the results were quite inefficient. ABC and PSOBOA are competitive in their convergence phenomenon, but the rank of PSOBOA is much higher than that of ABC.

The convergence curves of all these algorithms are shown in Fig. 4.2. It is observed in the convergence curve graph that PSOBOA performs better than other algorithms, especially its counterparts i.e. PSO and BOA. The transition from exploration to exploitation is quite smoothly exhibited by PSOBOA, however, the overall convergence time taken is more, as seen from the statistical analysis. This is possible because, as compared to the conventional algorithms, more parameters are processed in PSOBOA due to hybridization. Hence, additional calculations lead to the consumption of more time in solving the same problem.

Overall, PSOBOA was able to solve COPs effectively as can be seen from the results obtained in both the COP i.e., Pressure vessel design and welded beam design problem. In terms of efficiency, significance, and attaining global optima results, PSOBOA edged over most algorithms taken in this study. The convergence curve shows a smooth transition from exploration to exploitation within the search space. However, the convergence time taken is slightly higher in PSOBOA. Therefore, the performance of the algorithm in terms of convergence time needs to be improved in future work.

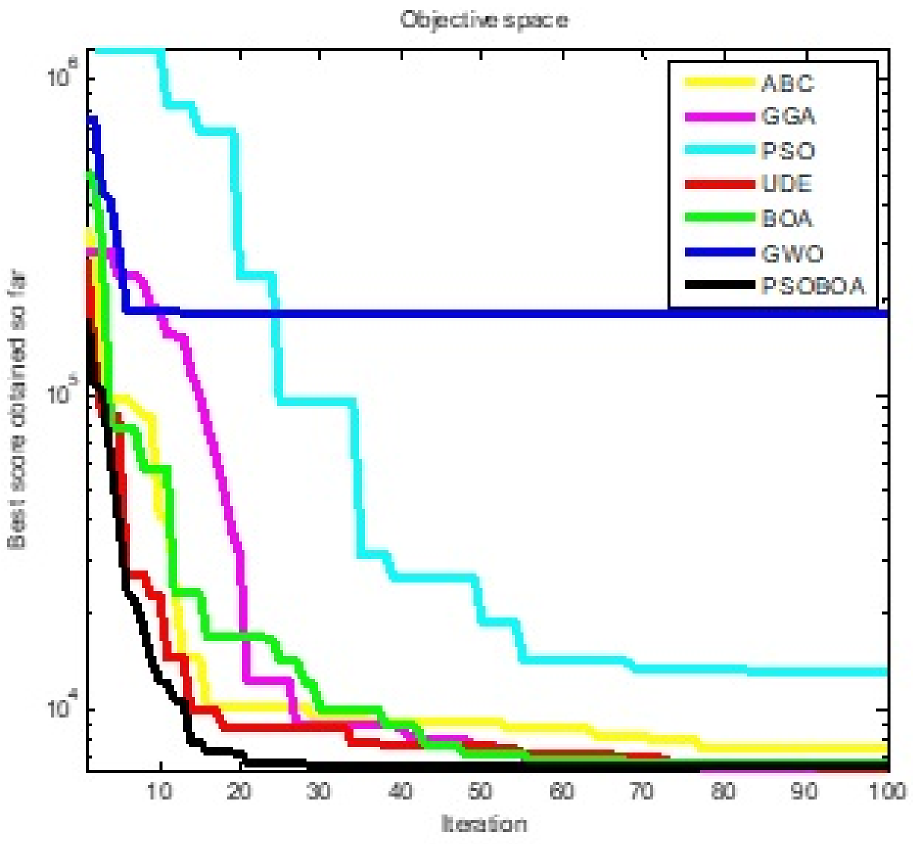


Figure 4.2: Convergence curves of different algorithms for welded beam design problem

4.5 Conclusion

This chapter proposes a novel hybrid of PSO and BOA by incorporating the key features of both PSO and BOA with three improvement strategies, as follows: (1) Redefining the objective function by incorporating a parameter-free penalty function. This simplifies the approach by converting the constrained problem into an unconstrained problem; (2) Use of the adaptive inertia weight and power exponent in the hybrid PSOBOA. This balances exploration and exploitation and makes the algorithm completely free from parameter-tuning; (3) Use of conditional approach in local and global phases of BOA to improve the convergence rate and avoid local optima trap.
To test the performance of the algorithm, it was subjected to two well-known COPs, namely pressure vessel design and welded-beam design problem, and the results were

compared with other well-known algorithms in the literature that were successfully used for COPs. The algorithm significantly outperformed others in terms of efficiency, local optima avoidance, exploration-exploitation balance, and reaching the global optimum. In the Friedman rank test, the proposed algorithm is statistically significant as compared to other algorithms.

Our new algorithm is efficient in solving real-world problems with constraints and tuning the parameters using an adaptive process. In the next chapter, we will focus on other techniques of tuning parameters of optimization algorithms and substantiating their suitability in other real-world problems.

CHAPTER 5

HYBRID CHAMELEON SWARM OPTIMIZATION-PARTICLE SWARM OPTIMIZATION ALGORITHM (CSAPSO)

An important consideration in evolutionary algorithms is testing their efficacy in real-world applications. In real-world problems, a lot of constraints are involved. However, to increase the accuracy of finding the best possible solution, it is always a good idea to increase the effective search space. For this, opposition-based learning plays an important role.

5.1 Introduction

Image Segmentation forms the basis of many real-world applications like computer vision, video applications, and medical imaging. In the context of medical imaging, the CXR of the patients who are ill with respiratory symptoms is indicative of pneumonia. Pneumonia causes an increase in lung density, which can be seen as whiteness in the lungs when examining the CXR. The degree of whiteness in CXR is directly proportional to the severity of pneumonia (134). Effective segmentation of CXR images into regions of interest (ROI) is crucial for accurate detection. Among different image segmentation techniques like thresholding, edge-detection, region-based, and graph-based (135; 136), the thresholding-based method is used widely due to its simple implementation (137; 138). The thresholding-based segmentation method divides the image into two (bi-level) or multiple (multi-level) classes based on the number of thresholds, where all the elements in a region share common properties like brightness, grey level, contrast, texture, color, etc. The Threshold segmentation method uses the gray probability and gray value to divide the image into different regions. Since the results rely purely on the threshold values, it's important to choose the right optimal value in segmentation. In multi-level thresholding, as the number of thresholds increases, the complexity of the model increases exponentially. Also, there are many challenges when performing thresholding, like, finding a proper objective function, applying the technique over the medical image, appropriate evaluation criteria, etc. Therefore, to produce a good quality segmented image, there is a need to deal with the discussed challenges.

In this work, we propose an efficient segmentation approach based on MCET, called CSAPSO, that utilizes the two algorithms at different stages to improve the convergence of CSA, namely; 1) OBL (139) and 2) PSO. OBL calculates the solution on the opposite side of the ordinary solutions and chooses the best solution for each search agent. Thus, it assists in improving the convergence of the algorithm to reach the global optima value. Afterward, PSO, which exploits the search space well, is used in parallel with the CSA to take care of the exploitation feature and avoid local optima, since CSA tends to give false optimal thresholds and high complexity due to the local optima trap. To avoid this, PSO is incorporated along with CSA to ensure an optimal balance between the exploration and the exploitation of the resultant hybrid algorithm. The implementation of the proposed methodology consists of three phases: 1) the Initialization and pre-processing phase, 2) the updating phase, and 3) the segmentation phase. In the initialization and pre-processing phase, OBL generates a random population of agents in the opposite direction. The cross-entropy function is then minimized to select the best solutions for the next phase. In the updating phase, the population is divided into unequal sub-populations and assigned to CSA and PSO in parallel to bring population diversity, overcome the potential local optima trap, and find the global optima value. The global optima value becomes the optimal threshold value. Finally, in the segmentation phase, the image is segmented using the optimal thresholds. The proposed CSAPSO algorithm demonstrates excellent performance in image thresholding, particularly in segmenting complex medical images. It outperforms CSA, PSO, and other classical and hybrid-based segmentation approaches. The performance evaluation includes comparisons using RMSE, Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity (SSIM) metrics, and the statistical analysis confirms the effectiveness of CSAPSO. Additionally, the algorithm's applicability is tested on a large dataset related to COVID-19, and comparisons are made with popular Data Learning Algorithms like U-Net (140), SegNet (141), and DeepLab (142). Overall, the CSAPSO algorithm provides an efficient and competitive image segmentation method, specifically for medical image analysis, with promising results for pneumonia detection and potential applications in other domains.
The main contributions of this work are as follows:

i A new method of multi-level image thresholding has been proposed based on MCET by evaluating the optimal threshold values based on opposition-based learning, CSA, and PSO.

ii The local and global search operators of both CSA and PSO are applied in parallel to the subpopulations, after refining the search space using OBL.

iii The performance has been tested on popular gray-scale images and real-time chest

X-ray images for the detection of pneumonia. Comparison is done with other state-of-the-art nature-inspired algorithms.

iv To test the robustness and generalization, the algorithm is further tested on COVID-19 images and compared with other Deep Learning Algorithms based on classification accuracy, area under the curve, etc.

5.2 Strategies employed in the CSAPSO algorithm

5.2.1 Minimum cross-entropy

The formulation of multi-level thresholding using minimum cross-entropy is presented here. Minimum cross-entropy was proposed by Kullback et al. (143). Consider two probability distributions, $P = p_1, p_2, ..., p_N$ and $Q = q_1, q_2, ..., q_N$. The cross-entropy between the two distributions is given as Eq. (5.1).

$$D(P, Q) = \sum_{k=1}^{N} p_k log(\frac{p_k}{q_k}) \tag{5.1}$$

where $D(P, Q)$ is the information-theoretic distance between the two distributions P and Q.
Li et al. (144) have implemented minimum cross-entropy in image segmentation for bilevel thresholding. Later, it extended to multi-level thresholding. Let the original image, I with gray levels L and histogram h. For multi-level thresholding with m number of thresholds namely th_1, th_2, th_3,... th_m, the cross-entropy is minimized between the original and segmented image. $m + 1$ classes are formed, namely, $C_0, C_1, C_2, \ldots, C_m$, where the intensity values range between $[0, th_1]$, $[th_1 + 1, th_2]$, .., and $[th_m + 1, L - 1]$ respectively. The multi-level segmented image is denoted as I' and defined as Eq. (5.2).

$$I' = \begin{cases} \mu_0, I(x, y) \leq th_1 \\ \mu_1, th_1 \leq I(x, y) \leq th_2 \\ . \\ . \\ \mu_m, I(x, y) \geq th_m \end{cases} \tag{5.2}$$

where μ_0, μ_1, μ_2,.., μ_m is the average mean of the classes $C_0, C_1, C_2, \ldots, C_m$, and is given as Eq. (5.3).

$$\mu_0 = \frac{\sum_{i=1}^{th_1} ih(i)}{\sum_{i=1}^{th_1} h(i)}, \mu_1 = \frac{\sum_{i=th_1+1}^{th_2} ih(i)}{\sum_{i=th_1+1}^{th_2} h(i)}, ...\mu_m = \frac{\sum_{i=th_m+1}^{L} ih(i)}{\sum_{i=th_m+1}^{L} h(i)} \tag{5.3}$$

For multi-level thresholding, the cross-entropy function is calculated as per Eq. (5.4).

$$\text{D(th}_1, th_2, ..., th_m) = \sum_{i=1}^{L} ih(i)log(i) - \sum_{i=1}^{th_1} ih(i)log(\mu_0) - \sum_{i=th_1+1}^{th_2} ih(i)log(\mu_1) - ... - \sum_{i=th_m+1}^{L} ih(i)log(\mu_m) \tag{5.4}$$

The first term of the equation is generally constant for each image; hence the function can be modified as per Eq. (5.5).

$$\text{D(th}_1, th_2, ..., th_m) = -\sum_{i=1}^{th_1} ih(i)log(\mu_0) - \sum_{i=th_1+1}^{th_2} ih(i)log(\mu_1) - ... - \sum_{i=th_m+1}^{L} ih(i)log(\mu_m) \tag{5.5}$$

The time complexity of the multi-level thresholding increases exponentially with the increase in the number of thresholds. Since the time complexity for bilevel thresholding is $O(L)$, the time complexity for multi-level thresholding would be $O(L^m)$.

5.2.2 Opposition-based learning mechanism

OBL was proposed by Tizhoosh in 2005 (139). The estimates and their counter-estimates form the basis of this model. Let $x \in [l, u]$ be a real number. Eq. (5.6) formulates the opposite number $\breve{x}$.

$$\breve{x} = l + u - x \tag{5.6}$$

This technique effectively improves the performance of the metaheuristic algorithms by enhancing population diversity. It deals with evaluating the current solutions as well as the opposition solutions along with their fitness functions. The better one participates in the next generation. This approach applies not only to the initial solution but to the subsequent generations of search agents as well. This is extremely helpful in cases when the current solution tends to go away from the optimal solution, especially when it is in the opposite direction.

5.3 Development of the CSAPSO algorithm

Due to stochastic behavior, there is always a scope for improvement in evolutionary algorithms. For the same reason, CSA sometimes fails to give optimum thresholds during the image segmentation process. To utilize its advantages to the fullest and avoid its limitations, we have improvised CSA in this paper by using two algorithms, namely the Opposition-based algorithm and the Particle swarm optimization algorithm, and therefore it is named CSAPSO. Our proposed method has three phases, namely; 1) Initialization and pre-processing phase, 2) updating phase, and 3) segmentation phase. These phases are described below in more detail.

5.3.1 Initialization and pre-processing phase

Initialization is an important step in an optimization algorithm since it determines the convergence of the algorithm and the final results. OBL strategy has proved its superiority in improving the convergence and quality of the solutions. The proposed CSAPSO algorithm begins by creating a population of threshold values X in dimensions n and N solutions. Eq. (5.7) gives the opposite solutions for the values.

$$\breve{X} = l + u - X \tag{5.7}$$

where u is the upper bound, and l is the lower bound of the search space. The minimum of X and $\breve{X}$ is selected for each search agent to find the best values among the population. This is shown in Eq. (5.8).

$$X_i = minimum(X_i, \breve{X}_i) \qquad \forall i = 1, 2, ..., N \tag{5.8}$$

After the best threshold values are selected for the initial population, the objective function i.e. minimum cross-entropy function is applied to them to evaluate the performance of these thresholds, using Eq. (5.9).

$$F(X_i) = -\sum_{i=1}^{X_1} ih(i)log(\mu_0) - \sum_{i=X_1+1}^{X_2} ih(i)log(\mu_1) - ... - \sum_{i=X_m+1}^{X_N} ih(i)log(\mu_m) \tag{5.9}$$

The search agent with the minimum cross-entropy value is considered the best agent. So, OBL boosts the convergence mechanism and achieves the global optima faster.

5.3.2 Updating phase

In this phase, two subpopulations are created from the population by applying the modulus function to the iteration number. If the current iteration is odd, the population is divided into $\frac{1}{3}^{rd}$ and $\frac{2}{3}^{rd}$ subpopulations. If the current iteration is even, the population is divided into $\frac{2}{3}^{rd}$ and $\frac{1}{3}^{rd}$ subpopulations. In either case, the first subpopulation is assigned to CSA and the second sub-population is assigned to PSO respectively and passed through the fitness function, i.e., the minimum cross-entropy function. The other fitness functions that are possible for image threhsolding are Otsu, Fuzzy entropy, Ranyi entropy, wavelet entropy, and Kapur's entropy function. However, in this paper, we use the minimum cross-entropy function since this function uses the mean values which remains steady and limited. So, it becomes easy to analyse the results and draw conclusions out of them. Now, the respective CSA and PSO algorithms update the position and velocity of all search agents. For the first subpopulation, i.e., the one with CSA, the position and velocity are updated using Eq. (5.10) and Eq. (5.11) respectively.

$$X_{t+1}^{i} = Xr_t^i + \overline{X}_t^i \tag{5.10}$$

$$\mathrm{V}_{t+1}^{i,j} = \begin{cases} V_t^{i,j} + p_1(local\_best_t^{i,j} - Global\_best_t^j)r_2 + p_2(Global\_best_t^j - V_t^{i,j})r_1 & r_i \geq Pp \\ V_t^{i,j} + \mu((u_j - l_j)r_3 + l_j^d)sgn(rand - 0.5) & r_i < Pp \end{cases} \tag{5.11}$$

where $Global\_best$ and $local\_best$ are the best global and local positions of the search agents respectively. Also, for the second subpopulation, i.e. the one with PSO, the velocity and the position are updated using Eq. (5.12) and Eq. (5.13) respectively.

$$V_i^{t+1} = wV_i^t + c_1 * r_1 * (local\_best_i - X_i^t) + c_2 * r_2 * (Global\_best - X_i^t) \tag{5.12}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{5.13}$$

We obtain a single set of solutions by merging two solutions. The process repeats till the termination condition. In this way, the solutions received using CSA and PSO are diverse and can be explored and exploited well. So, we can utilize their powerful search abilities by giving them a major and minor part of the population in an alternate way. This ensures population diversity since the population is updated in every run differently. By assigning an unequal population, we also prevent the candidate solutions from having almost the same values after every run. PSO ensures the local optima trap avoidance, and CSA retains high computational speed. In unison, both algorithms work on improving the segmentation results.

5.3.3 Segmentation phase

After specified iterations, or until the termination condition, the final position is considered the global optimal solution for the given problem. This global optima solution sets out the threshold values used in image segmentation. The solutions of the algorithm are the final threshold values. The original image uses these threshold values to form multiple segments based on the levels of thresholds. The segmentation of the image leads to dividing the image into multiple sections based on similar intensity values. The efficiency of segmentation is decided by computing the PSNR, SSIM, etc. The run-time depends on the number of iterations and the population size used for finding the optimal thresholds. the number of thresholds impacts the segmentation results and plays less role in deciding the run-time of the proposed algorithm.

5.4 Application of CSAPSO algorithm to segmentation of chest x-ray images for detection of pneumonia

The image segmentation using thresholding gives the classification of the image into different segments based on their similarity among them. The number of segments varies with different levels of thresholding. For distinct observations, we take a higher number of thresholds. This method proves effective for the chest x-ray images since the main criteria are to separate the affected areas of the chest from the normal ones. It could help diagnose pneumonia at an early stage and prevent the further growth of the disease. The steps for the implementation of the proposed algorithm for the segmentation of CXR images for the detection of pneumonia are shown in Algorithm 8.

Algorithm 8 Steps for CXR image segmentation using CSAPSO

```
1. Input the CXR image.
2. Compute the histogram.
3. Set the initial values of the parameters of the algorithms CSA and PSO.
4. Construct the population and its opposite population using OBL.
5. Select the best population based on the values.
6. Apply the objective function and calculate the fitness value.
7. Perform CSA and PSO on unequally divided subpopulations.
8. Perform position and velocity updates on each search agent and merge the final
   solution set.
9. When the termination condition is met, the final position is the best global optimal
   threshold value.
10. Segment CXR image based on the optimal thresholds.
```

The framework of the proposed CSAPSO algorithm for image thresholding using the concept of opposition-based learning, CSA, and PSO is shown in Fig. 5.1.

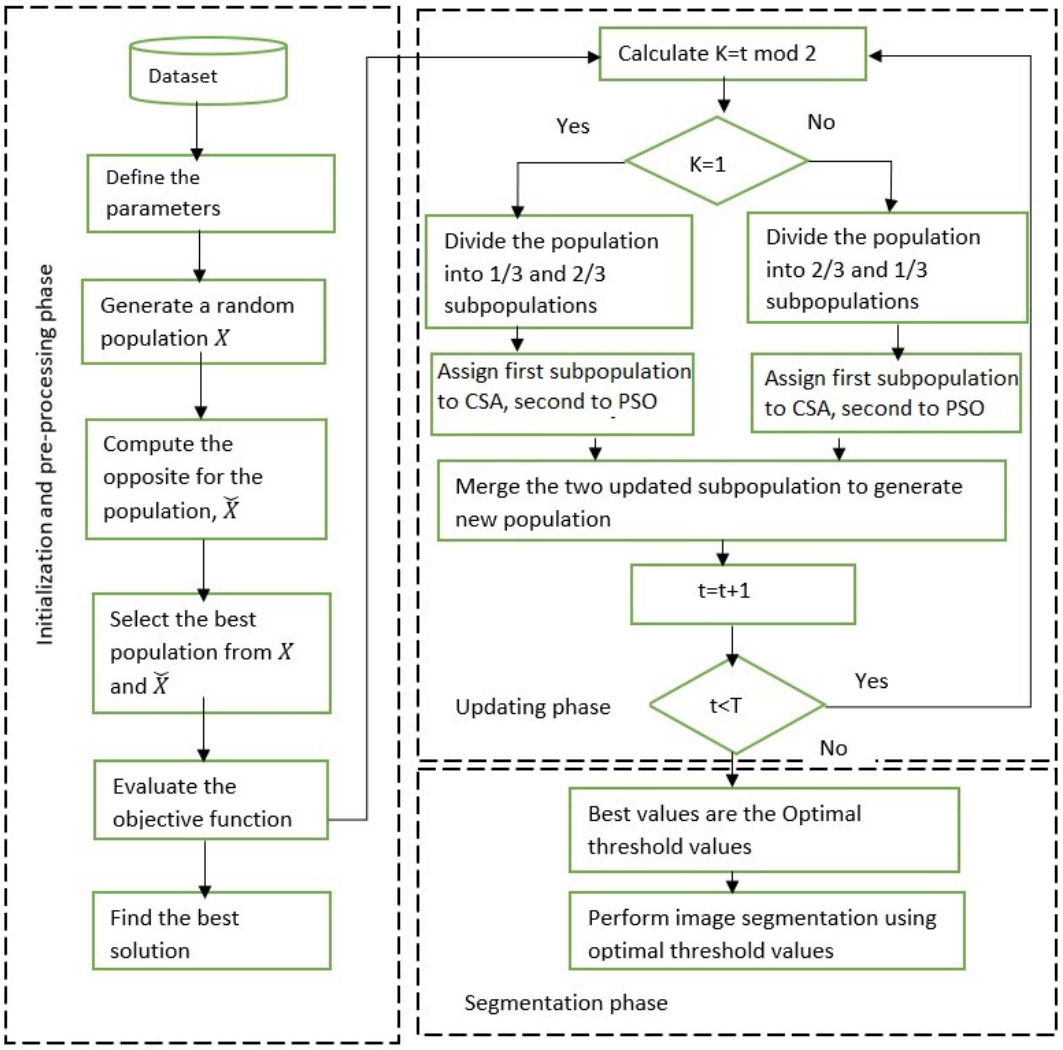

Dataset
Define the parameters
Generate a random population X
Compute the opposite for the population, $\breve{X}$
Select the best population from X and $\breve{X}$
Evaluate the objective function
Find the best solution
Initialization and pre-processing phase
Calculate K=t mod 2
Yes
No
K=1
Divide the population into 1/3 and 2/3 subpopulations
Divide the population into 2/3 and 1/3 subpopulations
Assign first subpopulation to CSA, second to PSO
Assign first subpopulation to CSA, second to PSO
Merge the two updated subpopulation to generate new population
t=t+1
Updating phase
t<T
Yes
No
Best values are the Optimal threshold values
Perform image segmentation using optimal threshold values
Segmentation phase

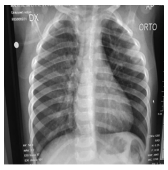
Figure 5.1: Framework of the proposed CSAPSO algorithm for image thresholding using the concept of opposition-based learning, CSA, and PSO

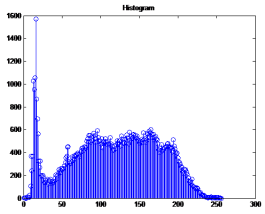
5.5 Experimental results and analysis

In this section, the performance of the proposed CSAPSO algorithm was tested on twelve chest x-ray images. The images were taken from (145) (146), curated by Paul Mooney, and are available on Kaggle. This dataset has 5,863 images in total, 4,387 of which are training images, 585 of which are validation images, and 1,091 of which are test images. The photos are in grayscale and have a 1,024 x 1,024 pixel resolution. The dataset is separated into the ”normal” and ”pneumonia” classes. In contrast to the ”pneumonia” class, which contains X-ray images of patients with pneumonia, the ”normal” class has images of healthy patients. These Chest x-ray images along with their histogram are shown in Fig. 5.2. The figure shows different characteristics of the images based on their histograms.

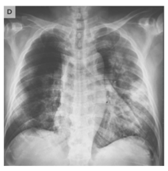
The comparison was done with ten algorithms consisting of some popular classical as well as hybrid metaheuristics, namely; CSA, PSO, Hybrid Particle Swarm Optimization Firefly Algorithm (HFPSO) (147), SSA(4), GA (148), Whale Optimization Algorithm (WOA) (149), GOA (150), Gravitational Search Algorithm-Genetic Algorithm (GSA-GA) (76), Salp Swarm Optimization-Moth Flame Optimization (SSAMFO) (151), Particle Swarm Optimization-Artificial Bee Colony-Ant Colony Optimization algorithm (PSO+ABC+ACO) (152). All the test images used for experimental evaluation were simulated thirty times using each multi-level thresholding algorithm. We experimented with the level=5, 10, 15, and 20, and the objective function to be minimized was the cross-entropy function. The results are analyzed visually as well as statistically. Visual results give the qualitative analysis and statistical results give the quantitative analysis. The experiments were done using Matlab.

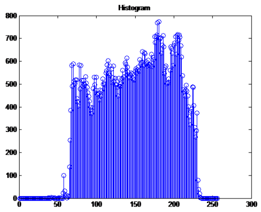
The parameter settings were the same for all the algorithms as mentioned in their respective papers. The number of candidate solutions considered in each iteration is based on the population size. The search space can be better explored with a larger population, but the computational cost may go up. New candidate solutions are produced from the current population using the mutation rate and crossover rate in CSA. The crossover rate regulates the likelihood of joining two solutions to produce a new one, whereas the mutation rate regulates the likelihood of each parameter in a solution being randomly modified. The balance between the particle’s previous velocity and the social and cognitive factors in the current iteration is controlled by the inertia weight. The attraction of the particle toward its personal best solution and the best solution discovered by the swarm are controlled, respectively, by the particle’s cognitive and social components. Depending on how difficult the problem is that needs to be addressed, different values for these parameters may be considered ideal. In both CSA and PSO, parameter tuning is frequently carried out by a trial-and-error process in which various parameter settings are evaluated on the problem at hand, and the best-performing one is chosen.


(a) I1

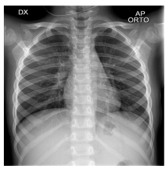


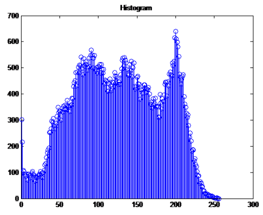
(b) Histogram of image I1


(c) I2

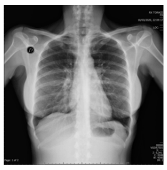


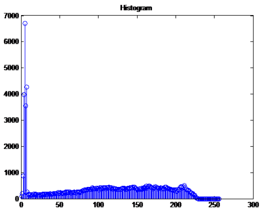
(d) Histogram of image I2


(e) I3

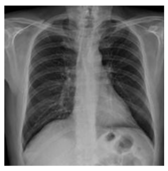


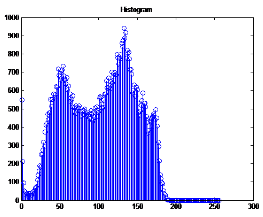
(f) Histogram of image I3


(g) I4

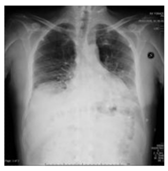


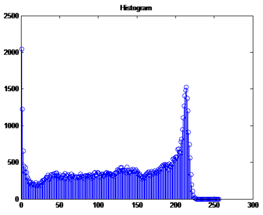
(h) Histogram of image I4


(i) I5

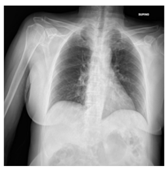


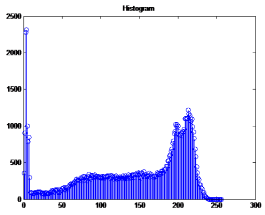
(j) Histogram of image I5


(k) I6

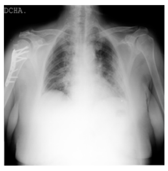


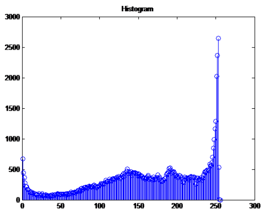
(l) Histogram of image I6


(m) I7

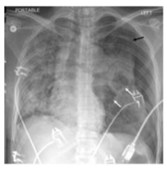


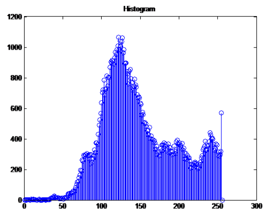
(n) Histogram of image I7


(o) I8

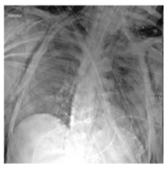


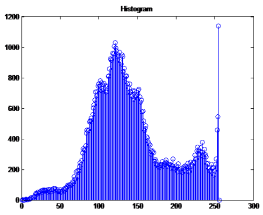
(p) Histogram of image I8


(q) I9

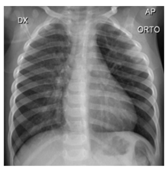


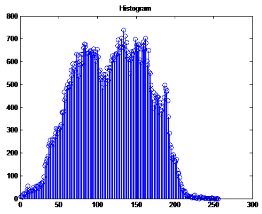
(r) Histogram of image I9


(s) I10

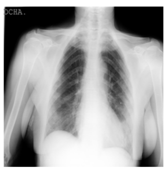


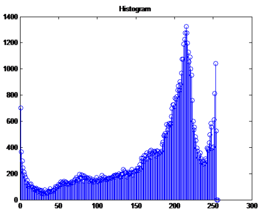
(t) Histogram of image I10


(u) I11



(v) Histogram of image I11


(w) I12



(x) Histogram of image I12

Figure 5.2: Original chest x-ray images and their histograms

The selection of parameters impacts how well these algorithms perform in contrast to other optimization algorithms like the SSA, GA, WOA, GOA, etc. Finding the ideal collection of parameters is essential for getting good results because each algorithm has its own set of variables that can be tuned. To choose the optimal algorithm for a particular optimization problem, it is important to carefully set the parameters of each method and compare their performances. For transparency and better result comparison, the two main parameters, i.e., population size and maximum iteration are set to 30 and 500, respectively for all the algorithms. The other parameter values for CSAPSO are mentioned as follows: -
$Pp = 0.1, \gamma = 1.0, \alpha = 3.5, \beta = 3.0, p_1 = 0.25, p_2 = 1.50, \rho = 1$

5.5.1 Evaluation methodology

The main aim of this paper is quality segmentation results and consistency. There are several parameters used to test the quality of segmentation. We have used the following parameters as performance indicators:

1. Root Mean Square Error (RMSE) works on the intensity values and gives the strength of the final constructed image, defined as per Eq. 5.14.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_r}(I_{i,j} - Is_{i,j})^2}{N_r * N_c}} \tag{5.14}$$

   where I and Is refers to the original and segmented images of size $N_r * N_c$.

2. Peak Signal-to-Noise ratio (PSNR) gives the accuracy of the final segmented image, defined as per Eq. 5.15.

$$PSNR = 20 * log_{10}(\frac{255}{RMSE}) \tag{5.15}$$

3. Structural Similarity Index (SSIM) gives the resemblance between original image and the thresholded image. It denotes the structural information degradation and is defined as per Eq. 5.16.

$$SSIM(I, Is) = \frac{(2\mu_I\mu Is + c_1)(2\sigma I, Is + c_2)}{(\mu_I^2 + \mu_{Is}^2 + c_1)(\sigma_I^2 + \sigma_{Is}^2 + c_2)} \tag{5.16}$$

   where μ_I and μ_{Is} are the mean intensities of I and Is, respectively, and σ_I and σ_{Is} are the standard deviations of the original image I and segmented image Is, respectively. The co-variance between I and Is is denoted by $\sigma_(I, Is)$. Constants c_1 and c_2 are set to 6.5025 and 58.52252, respectively (153).

Apart from these performance indicators, the Friedman rank-sum test at a 5% significance level was performed to see the difference between the proposed algorithm and other algorithms.

5.5.2 Performance analysis over general and COVID-19 CXR images

Fig. 5.3 shows the segmentation results on CXR images at threshold levels 5,10,15, and 20. The proposed CSAPSO algorithm successfully segmented the chest X-rays at various threshold levels. The results at threshold levels 10, 15, and 20 are better at threshold level 5. At each threshold level, the degree of whiteness in the X-rays is compared to normal chest X-rays to detect the severity of Pneumonia. As discussed, Pneumonia is identified by the degree of whiteness in the CXR. The proposed CSAPSO algorithm gave appropriate thresholds for segmentation as perceived visually from the figure.

Fig. 5.4 shows the segmented images for image I11 by CSAPSO and other algorithms used for comparison at threshold level 20. From the comparative analysis, it is observed that all the algorithms successfully segment the images. However, for the same threshold level (in this case, 20), CSAPSO gives better image quality as compared to other metaheuristics. The dark and light regions of the chest images are clearer. The thoracic area of the chest is accurately perceived, and the identification of affected areas in X-ray images becomes distinct. Hence, from the visual analysis, CSAPSO proves to be a better algorithm as compared to its counterparts and other state-of-the-art algorithms for the effective segmentation of the images.

To exhibit Pneumonia, two CXR images I5 and I11 were taken at threshold level 10 and segmented using CSAPSO as shown in Fig. 5.5. Both I5 and I11 are the CXR of persons affected by Pneumonia. The grey-colored section, denoted by the red arrow, is the affected area, whereas the black-colored section, denoted by the green arrow, is the unaffected area. The segmented image created using the thresholding technique displays the infected portions and the severity. From the figure, it is seen that there are multiple lesions or gray portions in image I11 that show how severe the disease is. Image I5 has fewer lesions due to the blacker portion and less grey portion, which shows the beginning of Pneumonia. Hence by segmenting the images using optimal threshold values, we can quickly know the impact of the disease on the lungs. However, the consideration is that the threshold levels chosen should be such that the affected and unaffected areas are distinctly visible. Hence, depending on the type of problem, we choose an appropriate threshold level.

To analyze the efficiency and accuracy of the proposed algorithm, fitness values, PSNR, SSIM, and CPU time are considered. Table 5.1 shows the fitness function values ob-

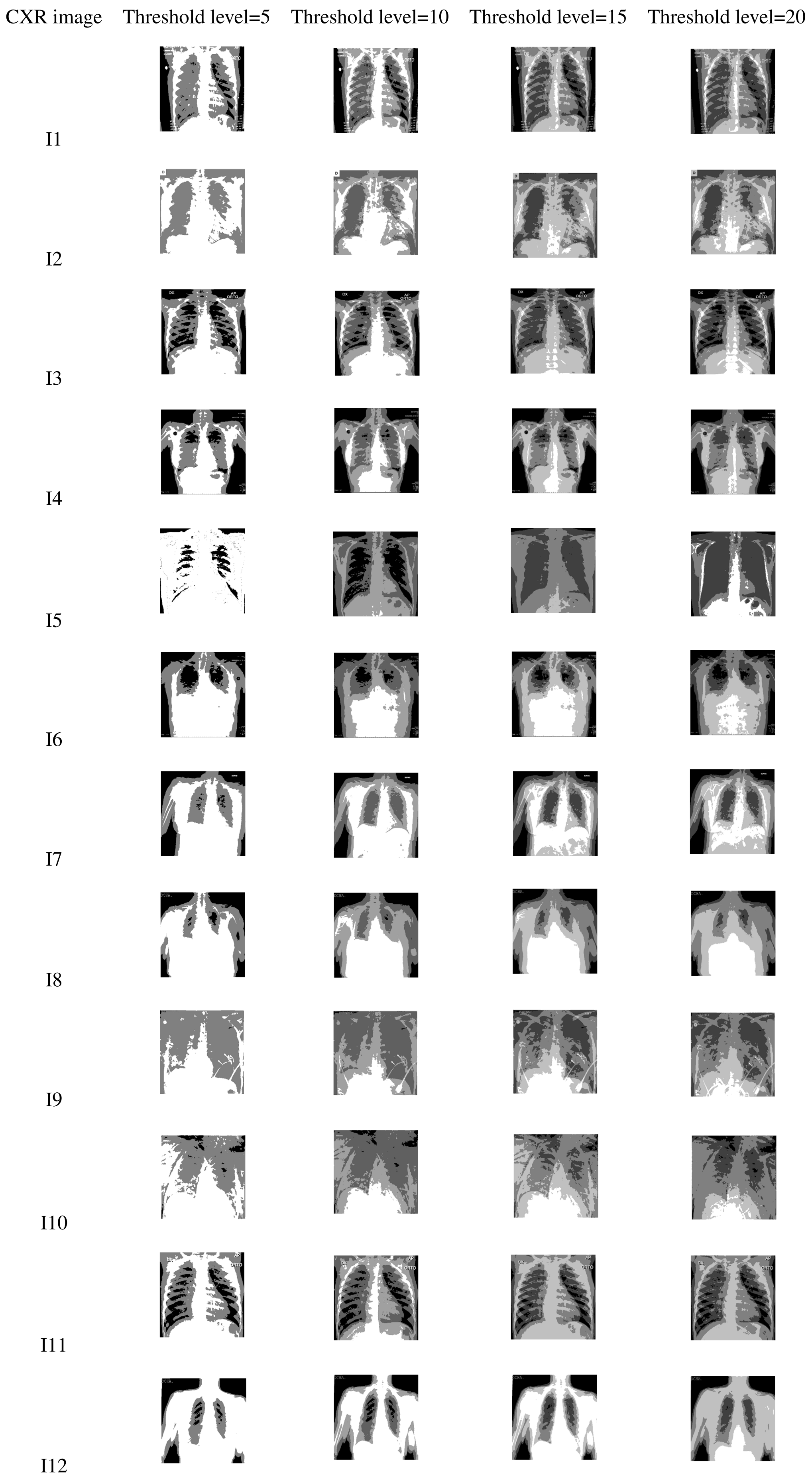


Figure 5.3: The segmented CXR images (I1-I12) using CSAPSO for threshold level 5,10,15, and 20

(a) CSAPSO

(b) PSO

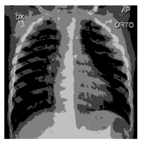
(c) HFPSO

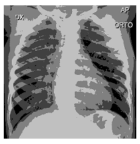
(d) SSA

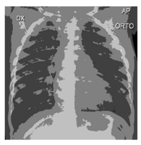
(e) GA

(f) WOA

(g) GOA


(h) GSA-GA


(i) SSAMSO


(j) PSO+ABC+ACO

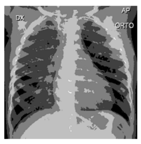
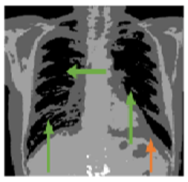

(k) CSA

Figure 5.4: Segmented CXR images obtained by each algorithm for image I11 at threshold level 20

tained using the proposed CSAPSO algorithm and other state-of-the-art algorithms. The best results are marked in bold. The fitness results give the quality of the threshold values and the accuracy of the results. It is observed from the table that CSAPSO stood first in delivering the best fitness values for most of the cases. The ranking was followed by SSAMFO and HFPSO. For some cases of lower threshold levels, SSAMFO gave better results than CSAPSO. The quality of the algorithm at higher dimensions is accessed using the higher threshold values. It is significant since real-world applications may contain several objects. One observation is that for higher threshold levels, the performance of CSAPSO was much better than at lower threshold levels. It shows that CSPSO proves to be a better optimizer than other algorithms for real-world applications.


(a) I5

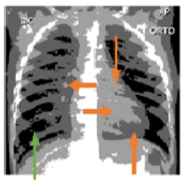

(b) I11

Figure 5.5: Illustration of pneumonia-affected and unaffected areas in CXR images I5 and I11 segmented using CSAPSO

Table 5.2 shows the results of the average PSNR results for all the images. From the detailed analysis of the performance at each threshold level, CSAPSO performs great for threshold levels 15 and 20. For threshold level 5, CSAPSO gave the best PSNR values for 4 cases, followed by GOA which gave the best values in 3 cases, followed by SSAMFO for 2 cases. For threshold level 10, CSAPSO gave the best PSNR values in 8 cases, followed by GOA for 4 cases. For threshold level 5, the performance of GOA and HFPSO was similar. However, it is outperformed by CSAPSO which gave the best values in 10 cases. Similar results were shown for threshold level 20, where the performance of GOA, SSAMFO, and PSO was the same, while CSAPSO outperformed others in 9 cases. Therefore, from the detailed analysis of the PSNR results, it can be seen that the performance of CSAPSO was the best, followed by GOA, and then SSAMFO. The other comparative algorithms like CSA, PSO, SSA, and WOA proved the worst for these cases. The impact of balanced exploration and exploitation can also be found in the quality of results. Since the local trap is avoided, therefore, the quality of results is significantly improved. Table 5.3 shows the average SSIM values for the images obtained using the proposed algorithm and other comparative algorithms. SSIM tells the similarity between the original image and the corresponding segmented image, hence it is an important quality parameter to validate the segmentation. The results were analyzed in detail and it was found that for threshold level 5, CSAPSO and SSAMFO were highly competitive and gave the best results in 6 and 5 cases respectively. For level 10, CSAPSO stood first with the best values in 6 cases, followed by SSAMFO and HFPSO. For level 15, CSAPSO gave the best results in 5 cases out of 12, followed by HFPSO, and then SSAMFO. The proposed approach gave the best results for 58% of cases in threshold level 20, and the rest of the best results were divided among SSAMFO (17%), GA (8%), SSA (8%), and HFPSO (8 %). Overall, it was observed that CSAPSO is competitive with SSAMFO for lower threshold levels. However, in totality, CSAPSO stood first, followed by SSAMFO, and then HFPSO. The performance of CSA is improved by using PSO and OBL. Fig. 5.6 shows the comparison of the average CPU taken by each algorithm for the first 5 CXR images over 30 runs and 500 iterations. It can be seen from the figure that the CPU time taken by CSAPSO was nearly 0.5 seconds, followed by PSO with 1 second execution time, SSAMFO which was nearly 1.3 seconds, and HFPSO with 1.5 seconds. SSA gave good results with nearly 3 seconds of execution. The algorithms like GA, GOA, and CSA took nearly 4 seconds of execution time. GSA-GA and WOA took around 5 seconds. The maximum CPU time was taken by PSO+ABC+ACO with nearly 6 seconds. The results of CDAPSO are achieved in less CPU time due to decreased complexity. Since PSO is a simple-to-implement algorithm, therefore when it is hybridized with CSA, it reduces the complexity of the algorithm and makes the function evaluations quicker. One more

Table 5.1: Fitness values obtained using each algorithm for the CXR images

Test images	Levels	CSAPSO	PSO	HFPSO	SSA	GA	WOA	GOA	GSA-GA	SSAMFO	PSO+ABC+ACO	CSA
I1	5	**16.84**	17.79	17.65	17.67	17.72	17.08	17.66	17.41	17.83	17.69	17.28
	10	**19.99**	20.65	20.81	20.66	20.28	20.32	20.14	20.07	20.46	20.87	20.15
	15	28.01	20.98	28.75	28.05	28.55	28.63	28.37	28.54	**27.77**	28.52	28.06
	20	31.88	31.12	**30.97**	31.96	31.24	31.62	31.06	31.65	31.27	31.29	31.83
I2	5	**17.99**	18.65	18.57	18.58	18.12	18.16	18.98	18.51	18.28	18.34	18.45
	10	22.03	22.78	**21.96**	22.09	22.91	22.73	22.59	22.18	22.06	22.13	22.52
	15	**26.87**	27.26	27.65	27.71	27.56	27.77	27.68	27.48	27.15	27.82	27.59
	20	**29.94**	30.25	30.64	30.92	30.48	30.83	30.79	30.45	30.47	30.34	30.86
I3	5	15.83	15.23	15.69	15.16	15.77	15.31	15.41	15.37	**14.86**	15.02	15.12
	10	**18.95**	19.27	19.89	19.12	19.21	19.24	19.63	19.91	19.36	19.72	19.28
	15	**26.95**	27.75	27.63	27.04	27.73	27.94	27.28	27.61	27.22	27.15	27.53
	20	**31.88**	32.04	32.87	32.45	32.27	32.41	32.11	32.56	32.14	32.19	32.12
I4	5	10.19	10.48	**9.87**	10.76	10.32	10.27	10.74	10.22	10.18	10.59	10.86
	10	**15.97**	16.74	16.35	16.83	16.22	16.25	16.79	16. 32	16.45	16.56	16.28
	15	**28.97**	29.53	29.11	29.15	29.34	29.78	29.67	29.52	29.68	29.22	29.18
	20	35.61	35.32	**34.95**	35.77	35.04	35.11	35.89	35.01	35.64	35.19	35.26
I5	5	16.43	**15.91**	16.83	16.12	16.72	16.88	16.39	16.09	16.11	16.84	16.49
	10	**23.97**	24.51	24.69	24.45	24.35	24.27	24.26	24.86	24.92	24.23	24.62
	15	**29.81**	30.31	30.44	30.67	30.11	30.25	30.67	30.47	30.17	30.29	30.56
	20	39.33	39.24	39.26	39.74	39.45	39.38	39.85	39.61	**38.92**	39.88	39.25
I6	5	**14.92**	15.13	15.22	15.72	15.84	15.46	15.63	15.85	15.55	15.17	15.51
	10	**19.96**	20.46	20.16	20.41	20.07	20.27	20.39	20.89	20.35	20.08	20.63
	15	28.49	**27.99**	28.37	28.34	28.15	28.24	28.76	28.85	28.31	28.29	28.67
	20	32.31	32.14	32.29	32.13	32.66	32.29	32.86	32.47	**31.95**	32.36	32.21
I7	5	**17.98**	18.11	18.33	18.52	18.94	18.83	18.77	18.57	18.09	18.29	18.39
	10	22.11	22.17	22.86	22.67	22.47	22.06	22.73	22.55	**21.89**	22.79	22.83
	15	**27.95**	28.46	28.24	28.74	22.16	22.29	22.94	22.55	22.68	22.96	22.71
	20	**33.72**	34.47	34.66	34.59	34.64	34.23	34.13	34.53	34.14	34.18	34.28
I8	5	**18.85**	19.08	19.57	19.11	19.73	19.33	19.36	19.68	19.06	19.29	19.29
	10	26.03	27.33	27.45	27.87	27.59	27.06	27.29	27.67	**27.95**	27.24	27.86
	15	**36.92**	37.23	37.19	37.51	37.25	37.72	37.28	37.67	37.01	37.24	37.87
	20	40.08	40.05	40.39	40.76	40.15	40.17	40.26	40.85	**39.97**	40.83	40.24
I9	5	**16.89**	17.35	17.38	17.53	17.05	17.24	17.48	17.23	17.61	17.13	17.41
	10	**18.88**	19.23	19.12	19.35	19.04	19.11	19.78	19.45	19.09	19.26	19.79
	15	**21.95**	22.79	22.57	22.33	22.14	22.57	22.49	22.12	22.18	22.45	22.69
	20	28.45	28.18	**27.97**	28.23	28.55	28.38	28.79	28.23	28.15	28.91	28.16
I10	5	20.58	20.01	20.55	20.29	20.11	**19.94**	20.42	20.23	20.31	20.17	20.38
	10	**26.97**	27.03	27.65	27.49	27.07	27.28	27.13	27.41	27.27	27.64	27.86
	15	**34.95**	35.08	35.56	35.13	35.38	35.15	35.23	35.11	35.29	35.74	35.44
	20	**39.99**	40.58	40.67	40.78	40.16	40.94	40.08	40.83	40.17	40.14	40.76
I11	5	**15.96**	16.84	16.05	16.58	16.12	16.17	16.44	16.23	16.52	16.63	16.31
	10	24.18	24.61	24.64	**23.87**	24.06	24.76	24.42	24.28	24.15	24.23	24.45
	15	**28.88**	29.59	29.22	29.07	29.16	29.16	29.01	29.22	29.38	29.27	29.55
	20	**31.99**	32.27	32.19	32.56	32.85	32.33	32.05	32.74	32.69	32.56	32.79
I12	5	**18.93**	19.36	19.27	19.55	19.68	19.21	19.17	19.78	19.03	19.88	19.75
	10	22.19	22.49	22.64	22.22	22.57	22.27	22.08	22.47	**21.96**	22.11	22.84
	15	**28.98**	29.97	29.33	29.81	29.64	29.32	29.35	29.17	29.48	29.43	29.82
	20	**40.95**	41.02	41.22	41.95	41.83	41.92	41.24	41.19	41.38	41.66	41.62

Table 5.2: Results of the PSNR values obtained using each algorithm for the CXR images

Test images	Levels	CSAPSO	PSO	HFPSO	SSA	GA	WOA	GOA	GSA-GA	SSAMFO	PSO+ABC+ACO	CSA
I1	5	**14.0628**	10.8354	13.9534	11.7352	12.2131	12.7482	13.1128	10.1836	12.7384	11.2848	11.1242
	10	**19.8747**	18.8356	16.2821	15.6273	15.4823	17.3293	17.8495	18.3342	17.2434	18.1243	17.1233
	15	**21.6753**	20.1729	20.1243	19.3826	18.3863	17.3864	19.1284	18.7239	18.2173	17.1294	19.3284
	20	**22.4947**	20.2361	21.2924	21.1287	20.2193	19.1282	21.1283	22.1923	20.2123	19.9875	19.3947
I2	5	6.0044	5.2328	8.2322	7.1294	4.2354	**10.3217**	3.2734	6.2353	9.3728	8.3758	8.3748
	10	19.4746	18.8273	18.2383	16.1739	15.0363	20.2381	**24.2739**	22.1836	19.9256	21.0982	22.1293
	15	21.931	20.9237	**23.2378**	20.9273	19.7389	18.8271	19.7637	18.7382	20.1294	21.1947	18.095
	20	20.5011	**24.8492**	22.6943	22.2848	21.1859	21.5826	19.9356	20.4537	19.8363	18.8537	19.7517
I3	5	**8.0064**	7.9204	7.8263	5.1523	6.284	6.1893	7.1729	6.284	7.2383	7.293	5.2398
	10	**19.9724**	11.2838	19.0009	18.3789	17.8365	12.7496	15.2398	13.2739	15.284	18.9475	17.8446
	15	**22.1094**	21.1813	21.3819	20.8361	19.9371	21.2817	22.0961	21.2819	20.1297	19.9172	19.6601
	20	**22.7307**	21.0128	20.1299	21.8123	20.1281	18.1289	17.1297	18.1289	19.128	20.1298	11.0372
I4	5	8.0182	10.9372	10.6384	10.1128	9.1738	5.1389	**11.979**	11.1238	10.1829	11.1289	5.5568
	10	**22.0849**	21.281	21.0119	20.0893	15.2173	17.4463	19.0263	18.9354	16.4583	18.3749	20.9371
	15	**21.6611**	21.0368	20.9364	20.1284	19.2179	18.1279	19.1278	20.1289	21.2379	20.1283	16.1289
	20	**21.7592**	20.1289	19.2178	18.1289	20.1289	20.9464	19.1289	20.1299	20.9623	20.9172	20.6152
I5	5	13.4419	10.9217	8.1289	9.8261	9.8127	12.8124	13.8217	14.1281	**14.9836**	11.8736	9.7362
	10	13.8327	15.9238	15.1289	12.1289	10.9217	12.9837	**16.0939**	14.3782	15.9462	15.8329	14.2367
	15	**19.1451**	18.8462	17.8372	17.3658	16.2178	16.3278	18.9438	18.4367	17.4832	17.9384	16.3742
	20	**17.8841**	16.7328	15.8543	15.7384	10.74328	13.3438	12.7388	11.9584	10.0043	12.8291	13.9321
I6	5	10.3729	10.8329	11.0942	10.8452	9.6839	9.2654	**11.4773**	10.7491	9.3376	8.2653	10.6382
	10	17.2738	18.9472	15.2378	15.9462	17.3682	17.2356	**19.339**	15.7294	16.9372	18.7378	19.0328
	15	**22.4181**	21.8923	22.0128	20.9832	19.7823	17.7237	17.0449	19.7823	20.0129	19.2389	18.3374
	20	**21.374**	20.0219	19.2189	19.4378	18.8329	17.7348	18.7438	19.4839	19.1193	18.7328	19.3278
I7	5	10.4958	11.7382	11.3829	10.8329	9.3728	10.3829	10.0992	11.8491	**12.7329**	9.5849	11.5748
	10	**17.1655**	16.9548	16.6237	15.8943	16.8473	17.0473	17.1431	16.8437	15.2479	16.6237	15.7238
	15	**23.0606**	21.3879	20.0573	20.8337	20.8462	19.3278	18.3782	18.0684	21.7492	22.8362	22.0582
	20	**21.9988**	20.0438	20.0582	19.9974	19.7294	18.3728	17.3759	16.34649	16.4826	19.3729	18.2749
I8	5	9.8493	8.3682	9.4378	9.2327	6.3482	8.2382	**10.738**	8.4389	8.3278	8.4389	9.9562
	10	**16.3492**	15.4832	13.9347	15.9582	15.8352	16.2857	15.2834	15.9274	15.9248	15.9284	16.10934
	15	**19.6393**	18.3283	18.3278	16.0678	18.0473	17.4926	19.5493	18.9326	19.1837	18.4337	17.3278
	20	18.4268	19.2857	18.3281	17.3659	18.9472	18.4652	17.4822	18.4825	**19.5135**	18.9362	19.1736
I9	5	**7.2372**	5.8731	6.0288	6.8803	5.9438	6.3278	7.5934	7.1107	7.3728	6.2301	7.1901
	10	**16.1167**	15.3728	15.8392	15.7382	16.0483	15.4378	15.9663	14.3735	15.9261	14.2368	15.9272
	15	**19.8327**	18.3628	18.3725	19.4382	19.1378	18.1389	18.9363	19.7462	18.37682	17.8366	18.3725
	20	**23.8867**	22.8563	22.4826	22.8741	23.6382	23.8462	22.4926	22.9472	21.9461	20.4392	20.3641
I10	5	**9.1063**	8.6472	8.9472	8.2834	9.0173	8.1268	8.3634	8.1635	8.9371	7.2718	6.1562
	10	18.2437	17.3728	17.1832	16.4735	16.5724	15.3619	**18.4707**	17.37619	16.3289	15.3826	15.3715
	15	18.3795	19.3867	19.3712	18.3712	20.0163	18.8462	**20.1308**	17.0562	16.9462	17.8526	18.8472
	20	**19.7059**	18.6381	19.0431	18.3728	18.8462	18.9371	19.4381	18.6482	18.4527	18.3268	17.3692
I11	5	**12.4645**	10.7362	11.9562	11.2545	10.2548	9.5282	10.5241	12.1835	11.7361	10.1356	11.5731
	10	**16.546**	16.2644	15.3108	15.0388	14.0931	14.0162	14.2673	13.6395	15.3687	15.3216	15.8534
	15	**18.7655**	17.3261	17.9654	17.6435	16.6432	16.3405	15.2549	14.8452	17.7352	17.3561	14.0351
	20	**22.0763**	21.8239	21.9463	20.7129	19.8351	21.9456	21.8831	21.5593	20.6691	21.8274	21.4374
I12	5	9.1733	8.6271	8.3627	8.2738	7.2562	8.56281	**9.6599**	8.1994	7.1763	7.237	8.3287
	10	**18.9142**	16.6342	17.6581	16.9473	16.5098	15.7459	16.6392	16.4262	17.6372	17.9435	18.2849
	15	**20.4071**	19.3487	19.3748	18.3728	18.3946	17.3657	18.3265	17.3267	17.9462	18.4825	17.3711
	20	**20.7885**	19.8462	19.3649	18.3906	18.4628	17.0462	17.4474	18.3628	17.9846	18.8361	19.8437

Table 5.3: Results of the SSIM values obtained using each algorithm for the CXR images

Test images	Levels	CSAPSO	PSO	HFPSO	SSA	GA	WOA	GOA	GSA-GA	SSAMFO	PSO+ABC+ACO	CSA
I1	5	**0.5058**	0.4378	0.4312	0.4639	0.4915	0.3715	0.4829	0.4871	0.4123	0.3821	0.2634
	10	0.6839	0.6283	0.6812	0.6245	0.6477	0.6245	0.5892	0.6182	**0.6934**	0.6528	0.6539
	15	0.7058	0.7037	**0.7894**	0.7363	0.7152	0.7637	0.7193	0.7473	0.7542	0.7079	0.7024
	20	**0.8947**	0.8827	0.8884	0.8007	0.8292	0.8057	0.8615	0.8633	0.8284	0.8259	0.8813
I2	5	0.4765	0.4623	0.4387	0.4225	0.4264	**0.4923**	0.4885	0.4916	0.4219	0.4657	0.4685
	10	**0.6913**	0.5528	0.5764	0.5025	0.5764	0.5174	0.5122	0.5172	0.6182	0.5336	0.5981
	15	**0.6987**	0.6364	0.6139	0.6036	0.6842	0.6652	0.6837	0.6208	0.6715	0.6759	0.6843
	20	0.7893	0.7665	0.7563	0.7126	**0.7943**	0.7677	0.7615	0.7201	0.7943	0.7842	0.7684
I3	5	0.5242	0.5061	0.5525	0.5308	0.5521	0.5582	0.5652	0.5224	**0.5934**	0.5226	0.5202
	10	**0.6937**	0.6784	0.6761	0.6768	0.6166	0.6777	0.6311	0.6876	0.6743	0.6527	0.6587
	15	0.7372	0.7803	0.7149	0.7505	0.7002	0.7226	0.7016	0.7544	**0.7951**	0.7783	0.7943
	20	**0.8934**	0.8458	0.8325	0.8177	0.8912	0.8687	0.8162	0.8292	0.8446	0.8422	0.8401
I4	5	**0.3969**	0.3002	0.3266	0.3142	0.3562	0.3774	0.3205	0.3422	0.3568	0.3476	0.3499
	10	0.4329	0.4539	**0.4964**	0.4804	0.4068	0.4182	0.4562	0.4775	0.4852	0.4049	0.4225
	15	0.5173	0.5268	0.5153	0.5182	0.5139	0.5073	0.5155	0.5698	**0.5741**	0.5525	0.5237
	20	0.6316	0.6121	0.6013	0.6544	0.6899	0.6139	0.6703	0.6748	**0.6987**	0.6561	0.601
I5	5	**0.4866**	0.4722	0.4663	0.4315	0.4145	0.4813	0.4703	0.4525	0.4317	0.4322	0.4474
	10	**0.5943**	0.5299	0.5736	0.5609	0.5365	0.5231	0.5465	0.5273	0.5468	0.5358	0.5317
	15	**0.6908**	0.6906	0.6115	0.6465	0.6172	0.6302	0.6434	0.6356	0.6562	0.6848	0.6207
	20	0.7745	0.7954	0.7565	**0.7956**	0.7006	0.7727	0.7664	0.7163	0.7115	0.7661	0.7435
I6	5	**0.5989**	0.5647	0.5208	0.5432	0.5743	0.5008	0.5054	0.5471	0.5817	0.5746	0.5671
	10	**0.6989**	0.6082	0.6101	0.6095	0.6187	0.6706	0.6832	0.6099	0.6204	0.6318	0.6955
	15	**0.7906**	0.7887	0.7895	0.7852	0.7001	0.7239	0.7263	0.7271	0.7266	0.7242	0.7154
	20	0.8211	0.8213	**0.8716**	0.8529	0.8263	0.8402	0.8332	0.8314	0.8416	0.8391	0.8088
I7	5	0.4299	0.4622	0.4805	0.4687	0.4741	0.4599	0.4201	0.4463	**0.4922**	0.4308	0.4897
	10	0.5577	0.5803	0.5093	0.5635	0.5149	**0.5856**	0.5644	0.5253	0.5366	0.5009	0.5153
	15	**0.5849**	0.5181	0.5386	0.5437	0.5301	0.5246	0.5129	0.5532	0.5329	0.5246	0.5336
	20	**0.7765**	0.7201	0.7502	0.7108	0.7302	0.7298	0.7695	0.7175	0.7481	0.7533	0.7135
I8	5	0.4382	0.4442	0.4279	0.4725	0.4174	0.4131	0.4708	0.4533	**0.4991**	0.4205	0.4831
	10	**0.5914**	0.5807	0.5103	0.5719	0.5276	0.5578	0.5269	0.5325	0.5363	0.5421	0.5568
	15	0.6628	0.6297	0.6366	0.6297	0.6202	0.6814	0.6207	0.6408	0.6807	**0.6998**	0.6191
	20	**0.7822**	0.7811	0.7155	0.7454	0.7152	0.7138	0.7814	0.7602	0.7429	0.7505	0.7596
I9	5	**0.4953**	0.4733	0.4368	0.4205	0.4172	0.4205	0.4763	0.4724	0.4836	0.4759	0.4625
	10	0.5109	0.5772	0.5665	0.5744	0.5394	0.5139	0.5485	0.5564	**0.5805**	0.5496	0.5579
	15	0.6124	0.6403	0.6805	**0.6957**	0.6444	0.6179	0.6198	0.6613	0.6051	0.6133	0.6446
	20	**0.7872**	0.7277	0.7306	0.7817	0.7152	0.7375	0.7274	0.7802	0.7673	0.7715	0.7335
I10	5	**0.5998**	0.5366	0.5287	0.5029	0.5907	0.5001	0.5502	0.5918	0.5295	0.5572	0.5925
	10	0.6215	0.6113	0.6516	0.6839	0.6627	0.6814	0.6167	0.6756	**0.6941**	0.6319	0.6396
	15	0.7782	0.7811	**0.7857**	0.7816	0.7412	0.7679	0.7104	0.7397	0.7619	0.7841	0.7273
	20	0.8395	0.8401	0.8363	0.8618	0.8234	0.8323	0.8179	0.8328	**0.8788**	0.8361	0.8274
I11	5	**0.3957**	0.3179	0.3234	0.3244	0.3359	0.3485	0.3285	0.3829	0.3369	0.3239	0.3328
	10	**0.4996**	0.4371	0.4219	0.4916	0.4782	0.4286	0.4957	0.4972	0.4835	0.4301	0.4731
	15	**0.5983**	0.5373	0.5978	0.5083	0.5514	0.5708	0.5077	0.5798	0.5947	0.5073	0.5419
	20	**0.6988**	0.6782	0.6388	0.6956	0.6671	0.6238	0.6073	0.6869	0.6144	0.6445	0.6627
I12	5	0.5431	0.5447	0.5365	0.5671	0.5579	0.5398	0.5712	0.5301	**0.5876**	0.5663	0.5498
	10	0.6124	0.6892	0.6399	**0.6983**	0.6255	0.6519	0.6204	0.6463	0.6291	0.6395	0.6291
	15	0.7313	0.7505	**0.7538**	0.7477	0.7305	0.7531	0.7345	0.7467	0.7176	0.7498	0.7485
	20	**0.8693**	0.8342	0.8477	0.8191	0.8279	0.8669	0.8691	0.8528	0.8357	0.8283	0.8242

interesting observation was that the graph of CPU execution time was almost a straight line which shows that the CPU time did not change much with the change in threshold levels. The results are the same with all the algorithms.

We tested the significance and efficiency of the proposed algorithm by using the fried-

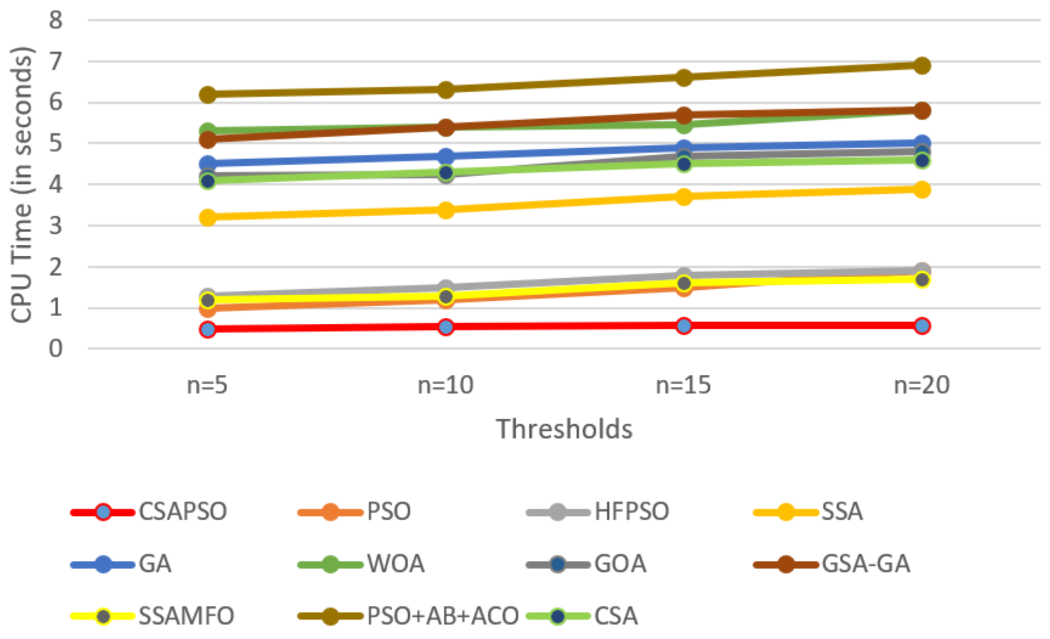


Figure 5.6: Comparison of average CPU time by each algorithm for first five images

man test. This test ranks the algorithms and studies the algorithm's robustness by using the median values. The higher the value, the better the rank. Also, the chief advantage is that it does not require the distribution of the dataset to be known. Table 5.4 presents the results of the Friedman rank-sum test. Our proposed approach obtained the best rank in the measures PSNR as well as SSIM, followed by SSAMFO and then HFPSO. The overall ranking of all the algorithms for both the measures is $CSAPSO > SSAMFO > HFPSO > PSO + ABC + ACO > PSO > CSA > SSA > GSA - GA > WOA > GOA > GA$. Therefore, CSAPSO proves to be statistically significant as compared to other algorithms for solving real-world problems.

Table 5.4: Results of the friedman test obtained using each algorithm for the CXR images

	CSAPSO	PSO	HFPSO	SSA	GA	WOA	GOA	GSA-GA	SSAMFO	PSO+ABC+ACO	CSA
PSNR	8.63	6.98	7.94	6.39	4.97	5.54	5.14	5.72	8.12	7.56	6.75
SSIM	7.28	6.19	6.97	5.93	3.42	5.13	3.95	5.52	7.01	6.68	6.05

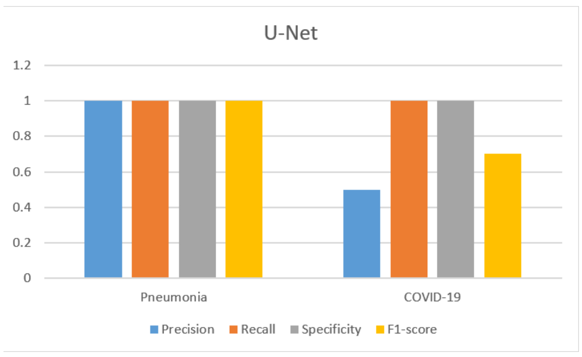
5.5.3 Performance analysis over deep learning classifiers

Another series of experiments was conducted on a large data set on COVID-19 [1]. In this series, the three deep learning models, namely U-Net (140), SegNet (141), and DeepLab (142) were trained over several parameters. Fig. 5.7 shows the graphical evaluation of all proposed X-ray image classifiers with four metrics of precision, recall, specificity, and f1-score. The positive cases of pneumonia and COVID-19 are confirmed while using U-Net and CSAPSO. The misclassification chances are more in the case of DeepLab and SegNet. So, from the figure, it can be concluded that the right classification is achieved in the case of U-Net and CSAPSO. Four expected outcomes of the confusion matrix are defined as follows by comparing the true labels of tested images and predicted outcomes of deep learning: The term True Positive (TP) demonstrated the accurate identification of chest diseases. A count of healthy instances is referred to as true negatives (TN). False positive (FP) refers to an error in which a test result falsely suggests the existence of a chest disease when the disease is not present. False negative (FN) refers to an error in which a test result fails to reveal the presence of a chest disease. From the results of the confusion matrix, the following five fundamental performance measures for deep image classifiers are as follows:
Accuracy: This measure evaluates the performance of the classifier. It is calculated as follows:
$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$
The precision recall or sensitivity, specificity, and f1-score are calculated as follows:
$Precision = \frac{TP}{TP+FP}$
$Recall = Senstivity = \frac{TP}{TP+FN}$
$Specificity = \frac{TN}{TN+FP}$
$F1 - Score = \frac{2(Precision X Recall)}{Precision+Recall}$

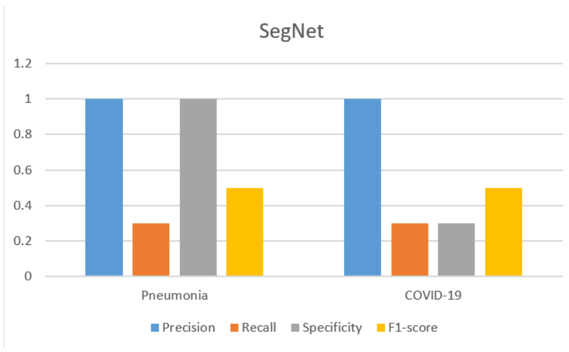
The comparative training and testing time analysis is shown in Fig. 5.8. The training time ranges from 900 seconds for CSAPSO to 7800 seconds for SegNet. On the other hand, the testing time ranges from 2 seconds for CSAPSO to 8 seconds for DeepLab. It shows an advantage of using CSAPSO with a minimum training time of 900 seconds and a testing time of 2 seconds. A higher training and testing time results in increased complexity.

Fig. 5.9 shows the testing accuracy and area under the curve for all the image classifiers. The best scores of accuracy and area under the curve are achieved by CSAPSO, i.e. 0.9, followed by U-Net. SegNet and DeepLab followed the performance of CSAPSO and U-Net. The tabular comparison of different deep learning classifiers is

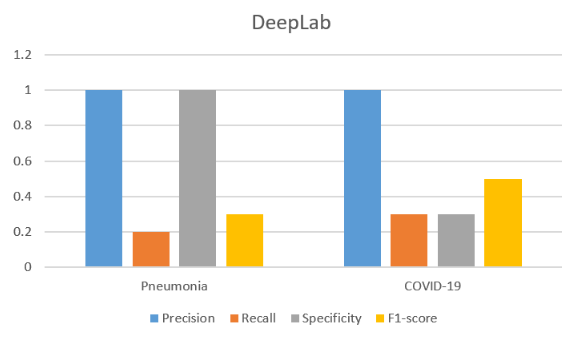
[1]The images were taken from https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711 and https://github.com/ieee8023/covid-chestxray-dataset



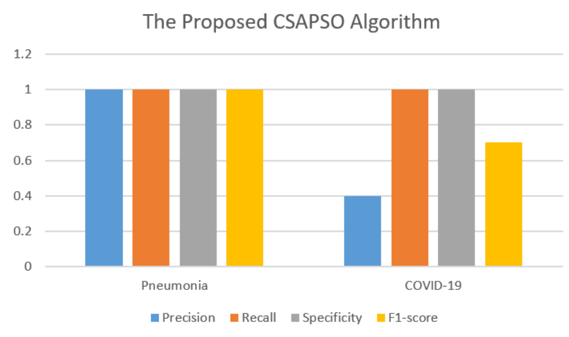
(a) U-Net



(b) SegNet

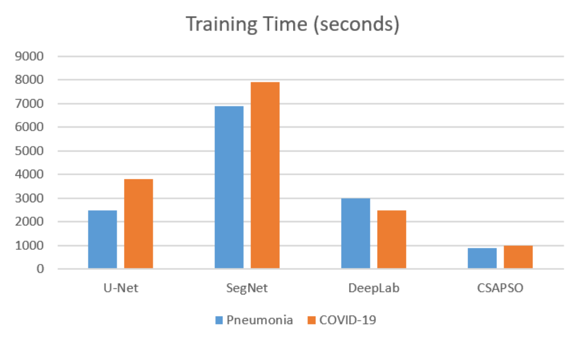


(c) DeepLab

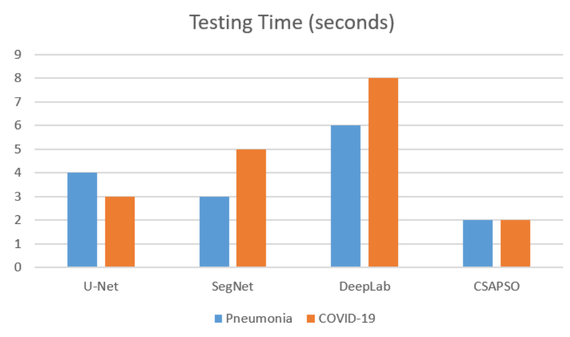


(d) CSAPSO

Figure 5.7: Evaluation results of deep learning classifiers to identify positive cases of pneumonia and COVID-19 diseases



(a) Training time



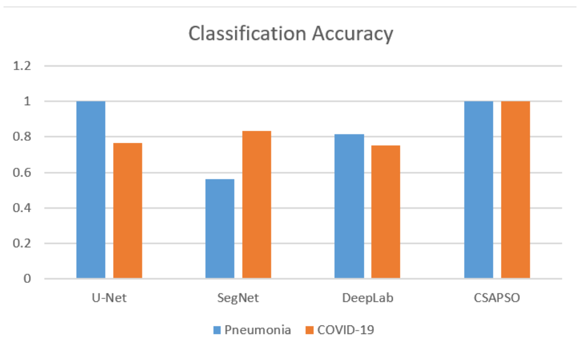
(b) Testing time

Figure 5.8: Computation times including training and testing times for the deep learning classifiers

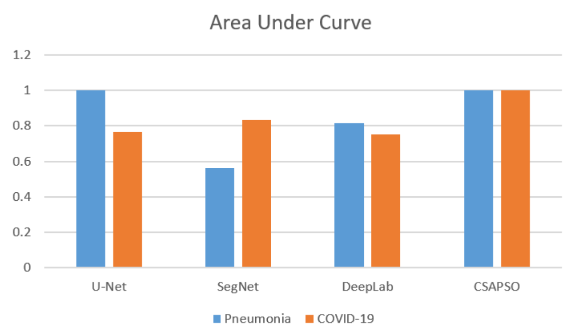
shown in Table 5.5. From the table, it is seen that U-Net and CSAPSO, both classify Pneumonia with 99.98% accuracy. However, when the dataset of COVID-19 is considered, CSAPSO outperforms U-Net with a classification accuracy of 99.93% whereas U-Net has an accuracy of 76.45%. The classifiers SegNet and DeepLab showed an average performance for both datasets.

Table 5.5: Comparative analysis of different deep learning classifiers

Classifier	Total Parameters(10ˆ6)	Trainable Parameters(10ˆ6)	Non-Trainable Parameters(10ˆ6)	Classification Accuracy	
				Pneumonia	COVID-19
U-Net	14.93	14.93	0	99.98	76.45
SegNet	20.76	20.71	0.05	56.14	83.23
DeepLab	58.12	58.1	0.02	81.34	75.12
CSAPSO	3.54	3.51	0.03	99.98	99.93



(a) Accuracy



(b) Area under curve

Figure 5.9: Classification accuracy and area under curve (AUC) for the deep learning classifiers

From the above series of experiments, we confirm the superiority of the proposed algorithm over other state-of-the-art algorithms. Our hybrid method, CSAPSO, combines the CSA, PSO, and OBL, allowing for quick convergence to the global optimum. This helps in reducing the amount of computation needed for image segmentation. CSAPSO utilizes the parallel execution of CSA and PSO on different sub-populations, which enhances optimization and improves computational efficiency. This ensures population diversity, which helps in avoiding local optima traps and produces superior segmentation outcomes. Medical image analysis is only one of the many image segmentation jobs that CSAPSO can be used for. It has been evaluated for the detection of pneumonia on CXR pictures, and it has demonstrated promising outcomes on COVID-19-related datasets. The Friedman test results proved that CSAPSO is statistically significant as compared to other algorithms. While our proposed method, CSAPSO, combines metaheuristic algorithms rather than deep learning techniques, however, our study also compared three deep learning techniques with our suggested method to segment medical images, and the results based on Accuracy, Precision, Recall, Specificity, and F-1 score, revealed that all four techniques were effective on the datasets we evaluated. There were some variations in performance, with CSAPSO outperforming U-Net, SegNet, and Deep Lab. Our approach does, however, have a few drawbacks that need to be addressed in further research. First, the existing architecture relies on time-intensive manual annotations for training. Second, a rather small dataset was used to train and test the suggested network, which would have limited its potential to generalize to other data sets or real-world events. Finally, the suggested method may not be appropriate for other medical imaging tasks because it is created specifically for the detection and localization of pneumonia in CXR images. The findings imply that deep learning methods for medical image segmentation show considerable promise, but further study is required to choose the most appropriate method for particular imaging modalities and applications. The study also emphasizes the need for larger data sets and standardized evaluation measures to enable reliable comparisons of various approaches.

5.6 Conclusion

In this work, we present the improvement of the CSA algorithm, named CSAPSO, through the integration of two other algorithms: PSO and OBL. The resultant algorithm, namely CSAPSO, is a multi-level thresholding-based image-segmentation method based on the minimum cross-entropy principle. CSAPSO minimizes the cross-entropy function by utilizing optimal threshold values obtained using CSA and PSO in a hybrid fashion. OBL is employed to enhance convergence towards the global optimal solution. Additionally, CSAPSO incorporates the division of the population into unequal sub-populations to ensure population diversity. The two sub-populations are run in parallel, with one using the CSA algorithm and the other utilizing the PSO algorithms. This allows the algorithm to leverage the global and local features of both algorithms simultaneously.

The proposed CSAPSO algorithm is implemented for the segmentation of twelve real-time CXR images and COVID-19 X-ray images. The experimental results are compared with state-of-the-art algorithms such as CSA, PSO, HFPSO, SSA, GA, WOA, GOA, GSA-GA, SSAMFO, PSO+ABC+ACO, as well as various deep learning algorithms including U-Net, SegNet, and DeepLab. Performance indicators such as RMSE, PSNR, SSIM, accuracy, and area under the curve are used for evaluation. The statistical significance of the results is tested using the Friedman rank test at a significance level of 5%.

The experimental results demonstrate that the proposed approach is highly competitive at the 5-threshold level and outperforms other algorithms at the threshold levels of 10, 15, and 20. The segmentation accuracy achieved, as indicated by the high PSNR and SSIM values, further highlights the effectiveness of the proposed method. The lower values of standard deviation indicate the stability of the algorithm, which is a crucial measure as many algorithms exhibit instability by producing different results with each iteration. The results of the Friedman rank-sum test also confirm the significance of CSAPSO in comparison to other methods. Moreover, the proposed algorithm successfully applies to real-life applications by producing optimal thresholds and generating high-quality segmented images of CXR for pneumonia detection. Therefore, it proves to be valuable in the field of medical imaging, enhancing the diagnostic performance of radiologists.

In future works, CSAPSO can be further utilized to optimize other image segmentation methods. Additionally, the development of a multiobjective version of the CSAPSO-based image thresholding technique could potentially improve its performance in segmenting CXR images.

The performance of the optimization algorithm is improved by performing various techniques like OBL. In the next chapter, improvement in the opposition-based EA is achieved using a Figurate strategy. A new framework for selective OBL is introduced to refine the quality of the solution and explore search space effectively.

CHAPTER 6

FIGURATE OPPOSITION-BASED PARTICLE SWARM OPTIMIZATION-GREY WOLF OPTIMIZER (Opp-PSOGWO)

The prime challenge behind different swarm-optimization approaches is to strike a proper balance between exploration and exploitation. The effectiveness of the algorithm comes when the search space is explored appropriately. Most of the existing algorithms use OBL for greater exploration features. However, it leads to an increase in computational complexity. This chapter aims to improve the search space exploration using Figurate OBL where the dimensions of particles that fall in the Fibonacci sequence positions are generated in opposite directions to achieve an optimal initial population with increased population diversity.

6.1 Introduction

The main idea behind different Swarm-based algorithms is to ensure a proper balance between exploration and exploitation while finding the global optimal value. However, according to (23), increasing exploration capability weakens exploitation capability and vice versa. The no-free lunch theory also establishes that no approach can effectively address all optimization problems. In this regard, GWO was developed by Mirjalili and is a recent meta-heuristic that takes inspiration from hunting techniques and the social hierarchy of grey wolves. It has better convergence in comparison to other metaheuristics. In addition, it has a better exploration-exploitation balance. There is a need to improve the convergence rate and strike an optimal exploration-exploitation balance. This motivates researchers to make improvements to the hybrid.
Based on the above analysis, this chapter aims to develop an efficient fusion of PSO and GWO, that is enhanced using Figurate Opposition-based Learning. Firstly, Figurate Opposition is used during initialization, where, the dimensions of the particles are selected on which the opposition is applied using OBL. To select the dimensions, the Fibonacci sequence is used which determines the position of particles that fall in the series, i.e., first, second, third, fifth, eighth positions, etc. This results in fast convergence and avoids unnecessary computation of all dimensions of particles. Second, a

new hybrid variant combining Figurate Opposition-based PSO and GWO is combined, named Opp-PSOGWO. The main idea is to enhance the exploitation of PSO and the exploration of GWO. This is done by updating the particle's position of the PSO particles using the position update equation of the wolves in GWO. The performance of the proposed method is demonstrated by testing it against nine standard benchmark functions and comparing the results with those of conventional PSO, GWO, and other comparative algorithms. The algorithm is tested for higher dimensions, which proves its stability. The analysis of convergence curves and the time required also reveals a clear distinction between the convergence of the proposed algorithm and that of other traditional methods.

6.2 Opposition based learning

OBL was proposed by Tizhoosh in 2005 (139) and is based on the estimates and its counter estimates. For higher dimensions, let $P(x_1, x_2, \dots, x_n)$ be a point in the n-dimensional space with $x_1, x_2, \dots, x_n$ as real numbers where each x_i lies in the range $[l_i, u_i]$. The opposite point, $\overset{v}{P}$ is defined in Eq. (6.1)

$$\overset{v}{x}_i = l_i + u_i - x_i \tag{6.1}$$

The advantages of this technique are discussed in Section 5.2.2.

6.3 Strategies employed in the Opp-PSOGWO algorithm

6.3.1 Figurate opposition-based learning

In this strategy, the opposite population is generated for the search agents using the Fibonacci sequence. This means that the opposition points for particles at positions 1,2,3,5,8,13,21,34….. and so on is selected. The candidates for new positions are chosen from the union of the present population and the opposite population based on the jumping rate J$_R$, or jumping probability. The jumping rate is set at 0.1 since a high jumping rate can distort the search phenomena and skip the actual solution to the problem. Additionally, the leaping rate provides a new way to determine the optimum and break from the local optimum. The jumping rate aids in this process when the optimum is in the opposite direction of the present solution, which happens sometimes. By using the Fibonacci sequence to define the positions, we avoid unnecessary calculations for all opposite positions. It quickens the search process and leads to global optima convergence quickly.

6.3.2 Integration of GWO

In the second strategy, GWO is integrated into PSO. The main idea is to initially update the position of search agents using PSO, which is further processed by the position update equation of GWO in the remaining iterations. In this way, the equations of GWO enhance the exploitation ability of PSO.

6.4 Development of the Opp-PSOGWO algorithm

In this section, Figurate opposition is done to the novel hybrid PSO-GWO to improve the convergence and achieve a better exploration-exploitation balance. Among the numerous state-of-the-art techniques, classical PSO is the most popular, since it is an easy algorithm to implement. However, from experimental analysis, it has been observed that it tends to fall into the local optima, which leads to premature convergence. Additionally, it has a low rate of convergence during iterations. To reduce this possibility, GWO is incorporated. To swiftly reach the global optimum, the exploration and exploitation capabilities of GWO and PSO, respectively, are coupled. The hybrid provides new advantages to the classical algorithm, along with the diversity of the newer problems to be resolved. We thereby propose an efficient hybrid by incorporating two strategies into PSO as follows:
Algorithm 9 presents the proposed methodology Opp-PSOGWO and Fig. 6.1 provides the associated flow chart. The process begins by initializing the particles in the search space at random. The fitness of each particle (or search agent) is assessed. A random value is generated by a random generator. Figurate opposition-based learning is used if the random value is smaller than the jumping rate; otherwise, GWO is used. Thus, the combination of Fibonacci-based opposition learning and GWO into PSO helps to increase diversity and avoid skipping real solutions.

6.5 Experimental results and analysis

The efficiency of the suggested OBL-PSOGWO algorithm has been demonstrated in several experiments. The other state-of-the-art algorithms used for comparisons are PSODE (24), PSOGA (25), and PSOGSA (26), along with its parent algorithms PSO (2) and GWO (5). The population size is set to 50, the maximum number of iterations is 500 and the algorithm is run independently 30 times on each benchmark function. The parameters of algorithms used for comparison are the same as those used in the literature. The experiments are carried out on MATLAB R2014a.
We use a collection of 9 well-known scalable problems with different difficulty levels, such as unimodal (F1-F3), multimodal (F4-F6) and fixed-dimensional multimodal functions (F7-F9). The sources for these problems are (154) and (149). Unimodal functions

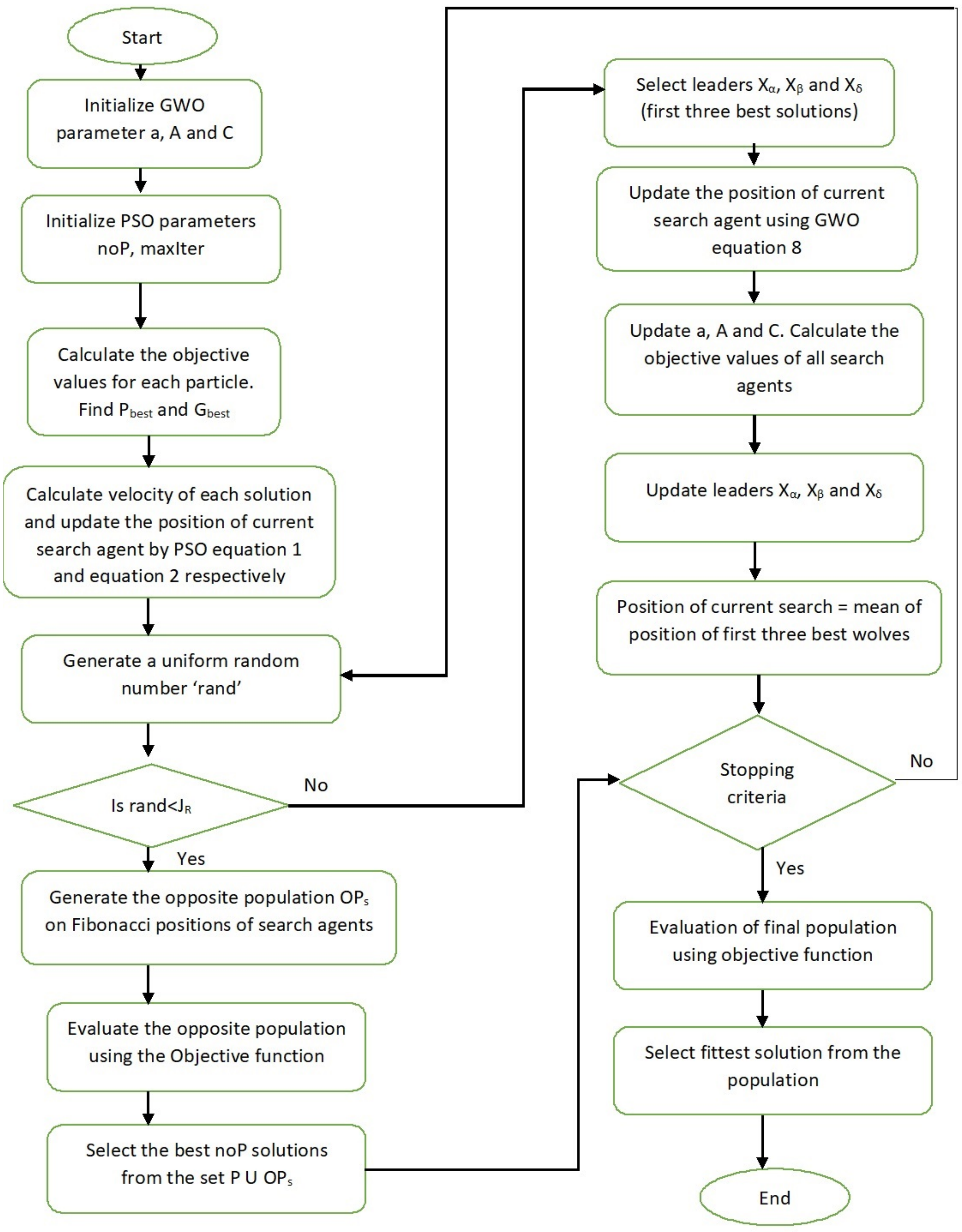

Start
Initialize GWO parameter a, A and C
Initialize PSO parameters noP, maxIter
Calculate the objective values for each particle. Find P_{best} and G_{best}
Calculate velocity of each solution and update the position of current search agent by PSO equation 1 and equation 2 respectively
Generate a uniform random number 'rand'
Is rand<J_R
No
Yes
Generate the opposite population OP_s on Fibonacci positions of search agents
Evaluate the opposite population using the Objective function
Select the best noP solutions from the set P U OP_s
Select leaders X_α, X_β and X_δ (first three best solutions)
Update the position of current search agent using GWO equation 8
Update a, A and C. Calculate the objective values of all search agents
Update leaders X_α, X_β and X_δ
Position of current search = mean of position of first three best wolves
Stopping criteria
No
Yes
Evaluation of final population using objective function
Select fittest solution from the population
End

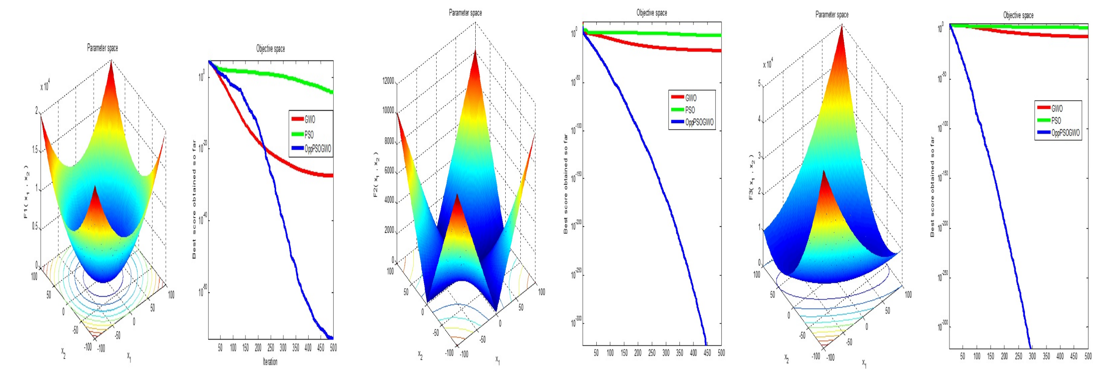
Figure 6.1: Flow chart of Opp-PSOGWO algorithm

Algorithm 9 The Proposed Opp-PSOGWO Algorithm

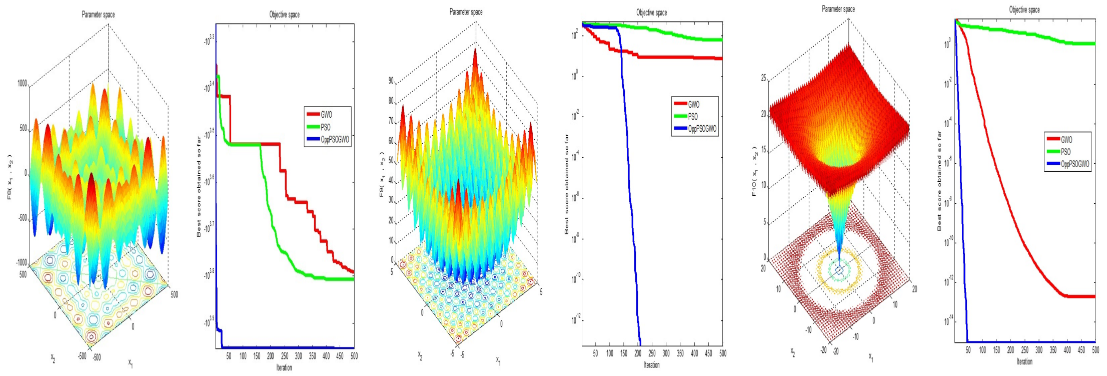
```
1. Initialize the swarm within the search space and the algorithm parameters- noP
   (number of particles), maxIter (maximum number of iterations), J$_R$ (jumping rate)
   =0.1.
2. Evaluate the initial fitness of each particle.
3. Update the $Pbest$ and $Gbest$.
4. Calculate the new velocity and position of the particles using Eq. (1.4) and Eq.
   (1.5), respectively.
5. While stopping criteria are met
6. Generate a random number rand.
7. If rand$<$ J$_R$.
8. The opposite population of solutions OP$_S$ is evaluated on the search agents at the
   Fibonacci positions i.e. 1,2,3,5,8, $\ldots$, etc.
9. Evaluate each solution of OP$_S$'s fitness.
10. Select the noP best solutions from the list of P U OP$_S$.
11. The best solution is updated.
12. Else
13. Select the values of $X_\alpha$, $X_\beta$ and $X_\delta$ as the first three best positions and apply
    GWO.
14. Calculate $X_1$, $X_2$ and $X_3$using Eq. (1.14) and update the positions of Grey
    wolves.
15. Update the values of a, A, and C.
16. Evaluate the fitness of the Grey wolves and update the leaders $X_\alpha$, $X_\beta$, and $X_\delta$.
17. Endif.
18. Calculate the new position of the prey using Eq. (1.15).
19. End While
20. Return the best global optima.
```

have a single optimum. These functions test the exploitation and convergence capability of the algorithm. The multimodal functions have several optima in which only one solution is the global optimal solution, the other solutions are the local optima. These functions test the exploration ability of the algorithm.
The qualitative analysis of the proposed algorithm is carried out by plotting the convergence curves of the Opp-PSOGWO algorithm and its parent PSO and GWO algorithms. Fig. 6.2 shows the curves for unimodal,multimodal, and composite benchmark functions. The proposed algorithm performs well in all cases. The curves demonstrate that, during the course of iterations, the proposed algorithm significantly increases the accuracy of the approximated optimum on the test functions. So, the proposed technique has merits in exploitation and exploration, respectively, and the performance on composite functions shows that the algorithm can solve complex functions with ease.

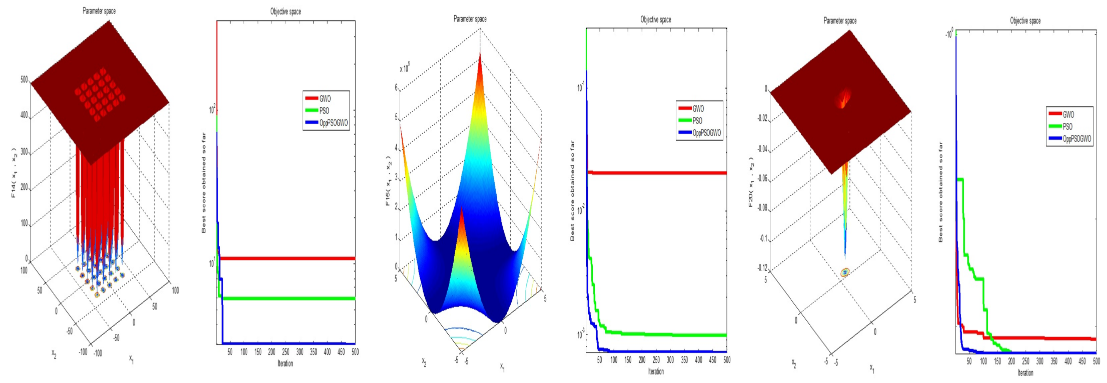
For quantitative analysis, the algorithm is run on 9 standard benchmark functions mentioned before. The mean, standard deviation (std) value, and CPU time are taken as evaluation criteria. The results are mentioned in Table 6.1. It is seen from the table that Opp-PSOGWO finds the best optimal solutions for unimodal functions, which shows that it has a strong exploitation ability as compared to other algorithms. Also, it outperforms the conventional PSO and GWO algorithms proving that it is better than its counterparts. For the multimodal functions, Opp-PSOGWO is highly competitive and gives the best result for almost all test functions. This helps us to conclude that the proposed solution can avoid the local minima trapping problem. For the functions (F7-F9), the dimension of the multimodal function is fixed, and the optima are more than one, so it requires a good amount of exploration and exploitation ability to find the solution. Opp-PSOGWO is found to provide the best optimal solution in every one of the composite situations. The lower std values show that the algorithm is stable in giving optimal values and the results obtained are almost the same in each run. However, in terms of CPU Time, the suggested approach is not dominating, as seen. The algorithm takes longer to run than other algorithms. This is due to the reason that opposition-based learning and GWO are introduced into a simple PSO algorithm. Increasing the search landscape using opposition-based learning leads to increased CPU time.
The scalability analysis of the proposed algorithm is done by increasing the limits of dimensions and testing them on the same benchmark functions. The results are shown in Table 6.2. The dimensions are set to 30, 50, and 100. The results of Opp-PSOGWO are competitive with its counterparts in some cases, like, F2, F8, and F9. However, it stands out in totality when considering the results for all the dimensions.
To perform the statistical analysis, we performed the Wilcoxon rank-sum test between Opp-PSOGWO and other comparative methods at a 5% significance level. The results



1(a) Search space of F1 1(b) Convergence curve of F1 2(a) Search space of F2 2(b) Convergence curve of F2 3(a) Search space of F3 3(b) Convergence curve of F3



4(a)Search space of F4 4(b) Convergence curve of F4 5(a)Search space of F5 5(b) Convergence curve of F5 6(a) Search space of F6 6(b) Convergence curve of F6



7(a)Search space of F7 7(b) Convergence curve of F7 8(a)Search space of F8 8(b) Convergence curve of F8 9(a) Search space of F9 9(b) Convergence curve of F9

Figure 6.2: 1-9(a) Search landscapes of function F1-F9 and 1-9(b) Convergence curves of the proposed Opp-PSOGWO algorithm and comparison with conventional PSO and GWO algorithms

Table 6.1: Results and comparison among different algorithms on standard benchmark functions

Function		PSO	GWO	PSODE	PSOGA	PSOGSA	Opp-PSOGWO
F1	Mean	4.054e-05	2.471e-28	1.389e-07	1.608e-08	6.023e-51	0.000e+00
	Time	1.37e-01	5.23e-01	8.43e-01	2.48e-01	9.32e-01	9.76e-01
	Std	4.72e-02	7.36e-02	2.42e-02	6.35e-03	1.32e-02	6.36e-01
F2	Mean	4.019e-03	6.852e-17	6.884e-06	1.005e-11	1.524e-47	0.000e+00
	Time	1.13e-01	1.38e-01	3.72e-01	2.91e-01	1.34e-01	2.47e-01
	Std	1.32e-03	4.85e-02	2.94e-03	6.67e-03	9.37e-02	3.85e-01
F3	Mean	7.585e-17	7.409e-08	1.093e-08	4.172e-23	5.347e-18	0.000e+00
	Time	2.31e-01	8.23e-01	3.23e-01	9.38e-01	7.85e-01	5.22e-01
	Std	5.32e-02	1.78e-02	4.65e-01	3.67e-03	9.78e-03	4.45e-01
F4	Mean	-3.073e+03	-6.047e+03	-2.686e+03	-1.463e+03	-7.073e+03	-8.234e+03
	Time	1.09e-01	2.83e-01	2.75e-01	2.95e-01	1.90e-01	2.61e-01
	Std	9.72e-02	1.65e-01	2.52e-02	6.35e-02	1.69e-02	6.45e-01
F5	Mean	45.7954	1.1369e-13	9.949e+00	1.404e+01	7.042e+00	0.000e+00
	Time	4.01e+00	2.42e+00	1.35e+00	1.71e+00	2.13e+00	2.45e+00
	Std	9.91e-02	7.25e-02	1.32e-03	8.36e-03	8.23e-01	2.13e-01
F6	Mean	7.692e-11	1.110e-13	1.6462e-03	2.995e-05	5.235e-10	8.882e-16
	Time	1.84e-01	7.23e-01	4.23e-01	2.19e-01	6.23e-01	6.88e-01
	Std	4.36e-01	3.16e-01	5.55.42e-01	1.42e-01	4.23e-02	1.43e-01
F7	Mean	1.992e+00	9.992e-01	1.992e+00	9.983e-01	1.831e+00	9.981e-01
	Time	1.26e-01	1.52e-01	1.02e-01	1.83e-01	1.55e-01	1.37e-01
	Std	1.72e-02	4.946e-02	3.48e-03	8.27e-03	3.29e-03	1.38e-01
F8	Mean	1.665e-04	3.861e-04	2.386e-04	8.699e-04	3.123e-05	7.521e-06
	Time	8.23e-02	2.48e-02	2.87e-02	3.66e-02	1.43e-02	4.19e-02
	Std	3.31e-01	2.42e-01	5.34e-01	1.31e-03	3.13e-02	6.24e-01
F9	Mean	-1.032e+00	-1.031e+00	-1.032e+00	-1.032e+00	-1.031e+00	-1.047e+00
	Time	1.36e-01	1.16e-01	4.24e-01	1.84e-01	1.86e-01	1.69e-01
	Std	1.342e-02	2.33e-02	2.09e-02	3.89e-03	9.93e-02	7.38e-01

are shown in Table 6.3. It is observed from the table that Opp-PSOGWO significantly outperforms other algorithms and, therefore, proves its applicability in real-world applications.

In conclusion, the suggested algorithm not only performs well in global searches but is also capable of avoiding local optima traps and producing superior outcomes. It significantly outperforms other comparative algorithms as well. Hence it proves that the fusion of Opposition-Based Learning with the hybrid of PSO and GWO is effective.

Table 6.2: Scalability analysis of the proposed Opp-PSOGWO and its comparison with counterparts

Function	Algorithm	Dim 30	Dim 50	Dim 100
F1	PSO	3.328e-53	1.733e+00	6.382e+00
	GWO	2.136e-55	6.382e-01	8.372e-01
	Opp-PSOGWO	0.00e+00	5.235e-01	8.235e-01
F2	PSO	7.384e-13	3.946e-27	9.362e-11
	GWO	5.327e-32	4.173e-32	7.435e-11
	Opp-PSOGWO	4.273e-49	8.125e-36	7.301e-27
F3	PSO	1.624e-27	3.173e-20	5.458e+00
	GWO	7.253e-30	9.236e-21	1.317e+00
	Opp-PSOGWO	6.236e-25	8.423e-18	1.943e+00
F4	PSO	-2.734e+16	-4.283e+15	-6.327e+12
	GWO	-6.270e+13	-2.128e+0e4	-7.638e+13
	Opp-PSOGWO	-7.123e+11	-2.439e+12	-5.274e+18
F5	PSO	7.638e-01	4.283e-01	2.183e-01
	GWO	3.823e-04	9.374e-11	7.378e+00
	Opp-PSOGWO	0.00e+00	1.373e-01	1.885e-01
F6	PSO	4.242e+01	8.632e+01	1.231e+01
	GWO	8.273e+01	3.128e-01	4.122e+01
	Opp-PSOGWO	0.00e+00	5.465e-15	7.348e-18
F7	PSO	2.314e-01	9.324e+00	6.345e+01
	GWO	2.488e-01	9.993e+00	4.234e+02
	Opp-PSOGWO	0.00e+00	1.673e-88	0.00e+00
F8	PSO	1.342e-08	8.435e-05	9.232e-11
	GWO	7.364e-11	3.478e-05	5.343e-10
	Opp-PSOGWO	1.399e-11	6.956e-08	8.994e-13
F9	PSO	3.545e+00	6.665e+00	4.345e+01
	GWO	7.348e-01	6.367e-01	2.372e-01
	Opp-PSOGWO	2.469e-01	1.975e-01	9.871e-02

Table 6.3: Wilcoxon rank-sum test of the proposed approach with other state-of-art techniques

	Opp-PSOGWO	
	H	p-value
PSO	28.0	0.000548297821
GWO	97.0	0.496322168437
PSODE	109.0	0.698465420658
PSOGA	82.0	0.184926374492
PSOGSA	54.0	0.063418399379

6.6 Conclusion

The proposed hybrid of PSO and GWO methods in this work combines the benefits of both algorithms. Opposition-based learning is integrated into the hybrid to prevent local optima traps and increase population diversity. A population is generated in the opposite direction in Fibonacci positions to achieve this. This fusion of opposition-based learning in the hybrid PSO-GWO improves the efficiency and exploration-exploitation balance of the search agents.
The experimental results are performed on 9 benchmark functions and the mean, standard deviation, and CPU time results are compared to other state-of-art methods. The results provide evidence that the proposed Opp-PSOGWO is better than its counterparts, as well as other related techniques. The proposed algorithm is scalable and gives good results on higher dimensions. Lastly, the Wilcoxon rank-sum test validates the proposed technique and proves its statistical significance. For the future scope, we will improve the technique for resolving multi-objective optimization issues in real scenarios.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Summary of the research work

To solve challenging real-world optimization problems, this thesis has concentrated on improving Evolutionary Algorithms. The study addressed the drawbacks of traditional algorithms and identified EA research gaps. BFAFA, PSOBOA, CSAPSO, and Opp-PSOGWO were presented as four unique hybrid algorithms and modifications to fill these gaps. Each algorithm was developed to enhance search capabilities, handle constraints, and improve optimization performance. On a variety of test challenges, including structural design, classification, and benchmark functions, the research was conducted through experimental assessments. The findings showed that the suggested algorithms performed better than the state-of-the-art algorithms in terms of convergence speed, solution quality, and robustness, significantly advancing the field of nature-inspired optimization.

7.2 Contributions and findings

In this thesis, the algorithms developed have been compared with existing state-of-the-art algorithms and shown significant results for different performance measures such as global optimum value, accuracy, mean, standard deviation, time, etc. The proposed algorithms perform well quantitatively and qualitatively. The development and validation of new hybrid algorithms and modifications to solve critical problems in EA are the research's main achievements.

- To improve the exploration of the search area, the hybrid of BFA and FA using the levy equation is developed. The performance is validated over standard, non-linear, and CEC_2017 functions, and compared with different state-of-the-art techniques that confirm the algorithm efficiency. The algorithm has been proved scalable over larger dimensions.

- To solve the Constrained Optimization Problem, we proposed a hybrid of PSO and BOA that uses self-adaptation, does not slacken the search process, and handles constraints by using a parameter-free penalty function. We have tested the proposed algorithm on structural optimization problems, such as pressure vessel design and welded-beam design problems, where the objectives, decision variables, and constraints are different. The experimental results and the convergence curves demonstrate better optimization performance of PSOBOA compared with state-of-the-art algorithms.

- Another hybrid is proposed to test the suitability of the optimization algorithm in general test problems. We have proposed a hybrid of CSA algorithm and PSO algorithm that combines OBL strategy and unequal division of population. To study the effectiveness, the proposed algorithm is used to investigate pneumonia by finding optimal segmentation of CXR of normal and COVID-19 patients. The performance metrics used are RMSE, PSNR, SSIM, accuracy, AUC, and Friedman rank test.

- The hybrid Opp-PSOGWO by combining PSO, GWO, and OBL is proposed that avoid the unnecessary calculations of OBL solutions by investigating the solutions on solutions at Fibonacci positions. To explore the search space effectively, OBL is applied to the hybrid of PSO and GWO. The proposed technique is compared with other state-of-the-art algorithms available in the literature on a variety of benchmark functions. The algorithm produces better results as compared to other algorithms. The results were compared in terms of efficiency, scalability, and reaching the global optimum.

7.3 Limitations and challenges

Despite its successes, this research had some restrictions and difficulties. The performance of the algorithms presented largely depended on the selection of their parameters, making it difficult to optimize the parameters for various domains of the problem. Additionally, the hybrid algorithms' computational complexity rose in comparison to some standard algorithms, which would limit their ability to be applied to complicated issues. Additionally, the experimental evaluations concentrated on particular problem domains, allowing room for additional research across a variety of other domains.

7.4 Future directions for research

- The future work will include extending the EA to multi-objective and binary problems.

- Extending EA to solve different real-world and nonlinear algorithms like biotechnology, risk management, process control, and environmental management.

- Improving the Convergence time of the hybrid-based EA with respect to its parent algorithms.

7.5 Conclusion

This research has successfully advanced optimization algorithms that are inspired by nature by suggesting new hybrid algorithms. The effectiveness and superiority of the suggested algorithms in solving challenging real-world situations have been confirmed by the experimental results. Although there are certain difficulties and restrictions, the research has established a strong framework for additional studies and advancements in this area. The developed algorithms and findings discussed in this thesis show promise for improving optimization methods in a variety of areas, giving academics and practitioners the tools they need to take on difficult real-world optimization tasks more successfully and effectively.

REFERENCES

[1] K. M. Passino, "Bacterial foraging optimization," in *Innovations and Developments of Swarm Intelligence Applications*, pp. 219–234, IGI Global, 2012.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[3] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[4] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Advances in engineering software*, vol. 114, pp. 163–191, 2017.

[5] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[6] S. Sharma, S. Mehta, and N. Chopra, "Economic load dispatch using grey wolf optimization," *International Journal of Engineering Research and Applications*, vol. 5, no. 4, pp. 128–132, 2015.

[7] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, vol. 23, pp. 715–734, 2019.

[8] M. S. Braik, "Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems," *Expert Systems with Applications*, vol. 174, p. 114685, 2021.

[9] N. Siddique and H. Adeli, "Nature inspired computing: an overview and some future directions," *Cognitive computation*, vol. 7, pp. 706–714, 2015.

[10] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced engineering informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[11] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[12] W. B. Langdon and R. Poli, *Foundations of genetic programming*. Springer Science and Business Media, 2013.

[13] Y. Cao and Q. Wu, “Evolutionary programming,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC’97)*, pp. 443–446, IEEE, 1997.

[14] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm intelligence*. Elsevier, 2001.

[15] M. Kumar, A. J. Kulkarni, and S. C. Satapathy, “Socio evolution and learning optimization algorithm: A socio-inspired optimization methodology,” *Future Generation Computer Systems*, vol. 81, pp. 252–272, 2018.

[16] S. Mishra, “A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 61–73, 2005.

[17] A. Khodabakhshian, E. Daryabeigi, and M. Moazzami, “A new optimization approach for multi-machine power system stabilizer design using a smart bacteria foraging algorithm,” *Simulation*, vol. 89, no. 9, pp. 1041–1055, 2013.

[18] D. H. Kim, A. Abraham, and J. H. Cho, “A hybrid genetic algorithm and bacterial foraging approach for global optimization,” *Information Sciences*, vol. 177, no. 18, pp. 3918–3937, 2007.

[19] M. M. M. Fuad, “A hybrid of bacterial foraging and differential evolution-based distance of sequences,” *Procedia Computer Science*, vol. 35, pp. 101–110, 2014.

[20] O. P. Verma, M. Hanmandlu, A. K. Sultania, and A. S. Parihar, “A novel fuzzy system for edge detection in noisy image using bacterial foraging,” *Multidimensional Systems and Signal Processing*, vol. 24, no. 1, pp. 181–198, 2013.

[21] T. Ya-lin, “Bacteria foraging optimization algorithm based on self-adaptative method,” *Value Engineering*, p. 11, 2015.

[22] A. E. Eiben and C. A. Schippers, “On evolutionary exploration and exploitation,” *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 35–50, 1998.

[23] M. S. Ahmad and A. Ahmad, “Hybrid pso-de technique to optimize energy resource for pv system,” *Int. J. Electr. Eng. Technol. IJEET*, vol. 12, pp. 128–139, 2021.

[24] R. K. Yadav *et al.*, “Pso-ga based hybrid with adam optimization for ann training with application in medical diagnosis,” *Cognitive Systems Research*, vol. 64, pp. 191–199, 2020.

[25] G. Eappen and T. Shankar, “Hybrid pso-gsa for energy efficient spectrum sensing in cognitive radio network,” *Physical Communication*, vol. 40, p. 101091, 2020.

[26] V. K. Kamboj, “A novel hybrid pso–gwo approach for unit commitment problem,” *Neural Computing and Applications*, vol. 27, no. 6, pp. 1643–1655, 2016.

[27] N. Singh and S. Singh, “Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance,” *Journal of Applied Mathematics*, vol. 2017, 2017.

[28] N. Chopra, G. Kumar, and S. Mehta, “Hybrid gwo-pso algorithm for solving convex economic load dispatch problem,” *Int J Res Adv Technol*, vol. 4, no. 6, pp. 37–41, 2016.

[29] I. Choulli, M. Elyaqouti, D. Saadaoui, S. Lidaighbi, A. Elhammoudy, I. Abazine, *et al.*, “Hybrid optimization based on the analytical approach and the particle swarm optimization algorithm (ana-pso) for the extraction of single and double diode models parameters,” *Energy*, vol. 283, p. 129043, 2023.

[30] W.-H. Tan and J. Mohamad-Saleh, “A hybrid whale optimization algorithm based on equilibrium concept,” *Alexandria Engineering Journal*, vol. 68, pp. 763–786, 2023.

[31] G. Liu, Z. Guo, W. Liu, F. Jiang, and E. Fu, “A feature selection method based on the golden jackal-grey wolf hybrid optimization algorithm,” *Plos one*, vol. 19, no. 1, p. e0295579, 2024.

[32] H. Chen, Y. Zhu, and K. Hu, “Self-adaptation in bacterial foraging optimization algorithm,” in *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, vol. 1, pp. 1026–1031, IEEE, 2008.

[33] T. Datta, I. Misra, B. Mangaraj, and S. Imtiaj, “Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence,” *Progress In Electromagnetics Research C*, vol. 1, pp. 143–157, 2008.

[34] R. Majhi, G. Panda, B. Majhi, and G. Sahoo, “Efficient prediction of stock market indices using adaptive bacterial foraging optimization (abfo) and bfo based techniques,” *Expert systems with applications*, vol. 36, no. 6, pp. 10097–10104, 2009.

[35] T. Ya-lin, “Bacteria foraging optimization algorithm based on self-adaptative method,” *Value Engineering*, p. 11, 2015.

[36] H. Chen, Y. Zhu, and K. Hu, "Cooperative bacterial foraging algorithm for global optimization," in *2009 Chinese control and decision conference*, pp. 3896–3901, IEEE, 2009.

[37] Q. Zhang, H. Chen, J. Luo, Y. Xu, C. Wu, and C. Li, "Chaos enhanced bacterial foraging optimization for global optimization," *Ieee Access*, vol. 6, pp. 64905–64919, 2018.

[38] W. Zhao and L. Wang, "An effective bacterial foraging optimizer for global optimization," *Information Sciences*, vol. 329, pp. 719–735, 2016.

[39] M. Issa, "Enhanced arithmetic optimization algorithm for parameter estimation of pid controller," *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 2191–2205, 2023.

[40] H. Song, J. Bei, H. Zhang, J. Wang, and P. Zhang, "Hybrid algorithm of differential evolution and flower pollination for global optimization problems," *Expert Systems with Applications*, vol. 237, p. 121402, 2024.

[41] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *Journal of computational design and engineering*, vol. 5, no. 4, pp. 458–472, 2018.

[42] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *2017 IEEE congress on evolutionary computation (CEC)*, pp. 1231–1238, IEEE, 2017.

[43] M. Z. bin Mohd Zain, J. Kanesan, J. H. Chuah, S. Dhanapal, and G. Kendall, "A multi-objective particle swarm optimization algorithm based on dynamic boundary search for constrained optimization," *Applied Soft Computing*, vol. 70, pp. 680–700, 2018.

[44] G. D'Angelo and F. Palmieri, "Gga: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems," *Information Sciences*, vol. 547, pp. 136–162, 2021.

[45] W. Gao, L. Huang, Y. Luo, Z. Wei, and S. Liu, "Constrained optimization by artificial bee colony framework," *IEEE Access*, vol. 6, pp. 73829–73845, 2018.

[46] J. Jelovica and Y. Cai, "Improved multi-objective structural optimization with adaptive repair-based constraint handling," *Engineering Optimization*, vol. 56, no. 1, pp. 118–137, 2024.

[47] H.-C. Lu, H.-Y. Tseng, and S.-W. Lin, "Double-track particle swarm optimizer for nonlinear constrained optimization problems," *Information Sciences*, vol. 622, pp. 587–628, 2023.

[48] Q. Kang, C. Xiong, M. Zhou, and L. Meng, "Opposition-based hybrid strategy for particle swarm optimization in noisy environments," *IEEE Access*, vol. 6, pp. 21888–21900, 2018.

[49] W. Long, J. Jiao, X. Liang, S. Cai, and M. Xu, "A random opposition-based learning grey wolf optimizer," *IEEE access*, vol. 7, pp. 113810–113825, 2019.

[50] S. Zhang, Q. Luo, and Y. Zhou, "Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method," *International journal of computational intelligence and applications*, vol. 16, no. 02, p. 1750012, 2017.

[51] R. A. Ibrahim, M. Abd Elaziz, and S. Lu, "Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization," *Expert Systems with Applications*, vol. 108, pp. 1–27, 2018.

[52] M. Gokuldhev, G. Singaravel, and N. Ram Mohan, "Multi-objective local pollination-based gray wolf optimizer for task scheduling heterogeneous cloud environment," *Journal of Circuits, Systems and Computers*, vol. 29, no. 07, p. 2050100, 2020.

[53] Y. Li, L. Jiao, R. Shang, and R. Stolkin, "Dynamic-context cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation," *Information Sciences*, vol. 294, pp. 408–422, 2015.

[54] A. Chatterjee, P. Siarry, A. Nakib, and R. Blanc, "An improved biogeography based optimization approach for segmentation of human head ct-scan images employing fuzzy entropy," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 8, pp. 1698–1709, 2012.

[55] Q. Abbas, M. T. A. Khan, A. Farooq, and M. E. Celebi, "Segmentation of lungs in hrct scan images using particle swarm optimization," *Int. J. Innov. Comput. Inf. Control*, vol. 9, no. 5, pp. 2155–2165, 2013.

[56] R. Panda, S. Agrawal, L. Samantaray, and A. Abraham, "An evolutionary gray gradient algorithm for multilevel thresholding of brain mr images using soft computing techniques," *Applied Soft Computing*, vol. 50, pp. 94–108, 2017.

[57] A. Ladgham, F. Hamdaoui, A. Sakly, and A. Mtibaa, "Fast mr brain image segmentation based on modified shuffled frog leaping algorithm," *Signal, Image and Video Processing*, vol. 9, no. 5, pp. 1113–1120, 2015.

[58] N. S. M. Raja, S. Fernandes, N. Dey, S. C. Satapathy, and V. Rajinikanth, "Contrast enhanced medical mri evaluation using tsallis entropy and region growing segmentation," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2018.

[59] Y. Li, X. Bai, L. Jiao, and Y. Xue, "Partitioned-cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation," *Applied Soft Computing*, vol. 56, pp. 345–356, 2017.

[60] R. Wang, Y. Zhou, C. Zhao, and H. Wu, "A hybrid flower pollination algorithm based modified randomized location for multi-threshold medical image segmentation," *Bio-medical materials and engineering*, vol. 26, no. s1, pp. S1345–S1351, 2015.

[61] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[62] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern recognition*, vol. 19, no. 1, pp. 41–47, 1986.

[63] F. Nie, P. Zhang, J. Li, and D. Ding, "A novel generalized entropy and its application in image thresholding," *Signal Processing*, vol. 134, pp. 23–34, 2017.

[64] J. N. Kapur, P. K. Sahoo, and A. K. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer vision, graphics, and image processing*, vol. 29, no. 3, pp. 273–285, 1985.

[65] Z. Jiang, F. Zou, D. Chen, and J. Kang, "An improved teaching–learning-based optimization for multilevel thresholding image segmentation," *Arabian Journal for Science and Engineering*, vol. 46, no. 9, pp. 8371–8396, 2021.

[66] S. Pare, A. K. Bhandari, A. Kumar, and G. K. Singh, "A new technique for multilevel color image thresholding based on modified fuzzy entropy and lévy flight firefly algorithm," *Computers and Electrical Engineering*, vol. 70, pp. 476–495, 2018.

[67] C. H. Li and C. Lee, "Minimum cross entropy thresholding," *Pattern recognition*, vol. 26, no. 4, pp. 617–625, 1993.

[68] A. K. Bhandari, A. Kumar, and G. K. Singh, "Tsallis entropy based multilevel thresholding for colored satellite image segmentation using evolutionary algorithms," *Expert systems with applications*, vol. 42, no. 22, pp. 8707–8730, 2015.

[69] T. Pun, "Entropic thresholding, a new approach," *Computer graphics and image processing*, vol. 16, no. 3, pp. 210–239, 1981.

[70] F. Nie, P. Zhang, J. Li, and D. Ding, "A novel generalized entropy and its application in image thresholding," *Signal Processing*, vol. 134, pp. 23–34, 2017.

[71] V. Rajinikanth and M. Couceiro, "Rgb histogram based color image segmentation using firefly algorithm," *Procedia Computer Science*, vol. 46, pp. 1449–1457, 2015.

[72] S. Pare, A. Kumar, V. Bajaj, and G. K. Singh, "An efficient method for multilevel color image thresholding using cuckoo search algorithm based on minimum cross entropy," *Applied Soft Computing*, vol. 61, pp. 570–592, 2017.

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

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

[75] A. Raj, G. Gautam, S. N. H. S. Abdullah, A. S. Zaini, and S. Mukhopadhyay, "Multi-level thresholding based on differential evolution and tsallis fuzzy entropy," *Image and Vision Computing*, vol. 91, p. 103792, 2019.

[76] G. Sun, A. Zhang, Y. Yao, and Z. Wang, "A novel hybrid algorithm of gravitational search algorithm with genetic algorithm for multi-level thresholding," *Applied Soft Computing*, vol. 46, pp. 703–730, 2016.

[77] M. Abd El Aziz, A. A. Ewees, and A. E. Hassanien, "Hybrid swarms optimization based image segmentation," in *Hybrid soft computing for image segmentation*, pp. 1–21, Springer, 2016.

[78] A. A. Ewees, M. Abd Elaziz, M. A. Al-Qaness, H. A. Khalil, and S. Kim, "Improved artificial bee colony using sine-cosine algorithm for multi-level thresholding image segmentation," *IEEE Access*, vol. 8, pp. 26304–26315, 2020.

[79] E. Tuba, M. Tuba, and D. Simian, "Support vector machine optimized by firefly algorithm for emphysema classification in lung tissue ct images," *Digital Library, University of West Bohemia*, 2017.

[80] N. S. M. Raja, P. V. Lakshmi, and K. P. Gunasekaran, "Firefly algorithm-assisted segmentation of brain regions using tsallis entropy and markov random field," in *Innovations in Electronics and Communication Engineering*, pp. 229–237, Springer, 2018.

[81] H. M. Ahmed, B. A. Youssef, A. S. Elkorany, A. A. Saleeb, and F. Abd El-Samie, "Hybrid gray wolf optimizer–artificial neural network classification approach for magnetic resonance brain images," *Applied optics*, vol. 57, no. 7, pp. B25–B31, 2018.

[82] S. Sarkar, S. Das, and S. S. Chaudhuri, "Multi-level thresholding with a decomposition-based multi-objective evolutionary algorithm for segmenting natural and medical images," *Applied Soft Computing*, vol. 50, pp. 142–157, 2017.

[83] D. Oliva, S. Hinojosa, E. Cuevas, G. Pajares, O. Avalos, and J. Gálvez, "Cross entropy based thresholding for magnetic resonance brain images using crow search algorithm," *Expert Systems with Applications*, vol. 79, pp. 164–180, 2017.

[84] D. Shao, C. Xu, Y. Xiang, P. Gui, X. Zhu, C. Zhang, and Z. Yu, "Ultrasound image segmentation with multilevel threshold based on differential search algorithm," *IET Image Processing*, vol. 13, no. 6, pp. 998–1005, 2019.

[85] L. Abualigah, K. H. Almotairi, and M. A. Elaziz, "Multilevel thresholding image segmentation using meta-heuristic optimization algorithms: Comparative analysis, open challenges and new trends," *Applied Intelligence*, vol. 53, no. 10, pp. 11654–11704, 2023.

[86] H. K. Bhuyan, V. Ravi, B. Brahma, and N. K. Kamila, "Disease analysis using machine learning approaches in healthcare system," *Health and Technology*, vol. 12, no. 5, pp. 987–1005, 2022.

[87] H. K. Bhuyan and V. Ravi, "Analysis of subfeature for classification in data mining," *IEEE Transactions on Engineering Management*, 2021.

[88] B. Brahma and H. K. Bhuyan, "Soft computing and machine learning techniques for e-health data analytics," in *Connected e-Health: Integrated IoT and Cloud Computing*, pp. 83–104, Springer, 2022.

[89] H. K. Bhuyan, T. A. Sai, M. Charan, K. V. Chowdary, and B. Brahma, "Analysis of classification based predicted disease using machine learning and medical things model," in *2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pp. 1–6, IEEE, 2022.

[90] D. H. Kim and J. H. Cho, "Adaptive tuning of pid controller for multivariable system using bacterial foraging based optimization," in *International Atlantic Web Intelligence Conference*, pp. 231–235, Springer, 2005.

[91] H. Chen, Q. Zhang, J. Luo, Y. Xu, and X. Zhang, "An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine," *Applied Soft Computing*, vol. 86, p. 105884, 2020.

[92] M. Tripathy, S. Mishra, L. L. Lai, and Q. Zhang, "Transmission loss reduction based on facts and bacteria foraging algorithm," in *Parallel Problem Solving from Nature-PPSN IX*, pp. 222–231, Springer, 2006.

[93] S. Banerjee and S. S. Chaudhuri, "Bacterial foraging-fuzzy synergism based image dehazing," *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 8377–8421, 2021.

[94] B. Niu, H. Wang, J. Wang, and L. Tan, "Multi-objective bacterial foraging optimization," *Neurocomputing*, vol. 116, pp. 336–345, 2013.

[95] M. Hanmandlu, O. P. Verma, N. K. Kumar, and M. Kulkarni, "A novel optimal fuzzy system for color image enhancement using bacterial foraging," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2867–2879, 2009.

[96] O. P. Verma, M. Hanmandlu, P. Kumar, S. Chhabra, and A. Jindal, "A novel bacterial foraging technique for edge detection," *Pattern recognition letters*, vol. 32, no. 8, pp. 1187–1196, 2011.

[97] O. P. Verma, M. Hanmandlu, A. K. Sultania, and A. S. Parihar, "A novel fuzzy system for edge detection in noisy image using bacterial foraging," *Multidimensional Systems and Signal Processing*, vol. 24, no. 1, pp. 181–198, 2013.

[98] O. P. Verma and A. S. Parihar, "An optimal fuzzy system for edge detection in color images using bacterial foraging algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 1, pp. 114–127, 2016.

[99] O. P. Verma, R. Sharma, and D. Kumar, "Binarization based image edge detection using bacterial foraging algorithm," *Procedia Technology*, vol. 6, pp. 315–323, 2012.

[100] K. Tang, X. Xiao, J. Wu, J. Yang, and L. Luo, "An improved multilevel thresholding approach based modified bacterial foraging optimization," *Applied Intelligence*, vol. 46, no. 1, pp. 214–226, 2017.

[101] L. Tan, F. Lin, and H. Wang, “Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows,” *Neurocomputing*, vol. 151, pp. 1208–1215, 2015.

[102] B. Hernández-Ocana, E. Mezura-Montes, and P. Pozos-Parra, “A review of the bacterial foraging algorithm in constrained numerical optimization,” in *2013 IEEE congress on evolutionary computation*, pp. 2695–2702, IEEE, 2013.

[103] M. Kowsalya *et al.*, “Optimal size and siting of multiple distributed generators in distribution system using bacterial foraging optimization,” *Swarm and Evolutionary computation*, vol. 15, pp. 58–65, 2014.

[104] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, “Synergy of pso and bacterial foraging optimization—a comparative study on numerical benchmarks,” in *Innovations in hybrid intelligent systems*, pp. 255–263, Springer, 2007.

[105] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[106] Q. Zhou and Y. Zhou, “Wolf colony search algorithm based on leader strategy,” *Application Research of Computers*, vol. 30, no. 9, pp. 2629–2632, 2013.

[107] D. Wang, X. Qian, K. Liu, X. Ban, and X. Guan, “An adaptive distributed size wolf pack optimization algorithm using strategy of jumping for raid (september 2018),” *IEEE Access*, vol. 6, pp. 65260–65274, 2018.

[108] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[109] M.-W. Li, Y.-T. Wang, J. Geng, and W.-C. Hong, “Chaos cloud quantum bat hybrid optimization algorithm,” *Nonlinear Dynamics*, vol. 103, no. 1, pp. 1167–1193, 2021.

[110] S. K. Pal, C. Rai, and A. P. Singh, “Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems,” *International Journal of intelligent systems and applications*, vol. 4, no. 10, p. 50, 2012.

[111] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, and X. Yao, “Benchmark functions for cec’2017 competition on evolutionary many-objective optimization,” in *Proc. IEEE Congr. Evol. Comput.*, pp. 1–20, 2017.

[112] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization," *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

[113] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[114] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural computing and applications*, vol. 31, no. 7, pp. 1995–2014, 2019.

[115] P. Kora, A. Abraham, and K. Meenakshi, "Heart disease detection using hybrid of bacterial foraging and particle swarm optimization," *Evolving Systems*, vol. 11, no. 1, pp. 15–28, 2020.

[116] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley and Sons, 2013.

[117] Z. Zhang and W.-C. Hong, "Application of variational mode decomposition and chaotic grey wolf optimizer with support vector regression for forecasting electric loads," *Knowledge-Based Systems*, vol. 228, p. 107297, 2021.

[118] H. Chickermane and H. C. Gea, "Structural optimization using a new local approximation method," *International journal for numerical methods in engineering*, vol. 39, no. 5, pp. 829–846, 1996.

[119] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: a new metaheuristic optimization algorithm," *Computers and Structures*, vol. 139, pp. 98–112, 2014.

[120] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with computers*, vol. 29, no. 1, pp. 17–35, 2013.

[121] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, 2008.

[122] H. Chickermane and H. C. Gea, "Structural optimization using a new local approximation method," *International journal for numerical methods in engineering*, vol. 39, no. 5, pp. 829–846, 1996.

[123] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Information Sciences*, vol. 178, no. 15, pp. 3043–3074, 2008.

[124] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, “Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems,” *Applied Soft Computing*, vol. 13, no. 5, pp. 2592–2612, 2013.

[125] T. Ray and P. Saini, “Engineering design optimization using a swarm with an intelligent information sharing among individuals,” *Engineering Optimization*, vol. 33, no. 6, pp. 735–748, 2001.

[126] J.-F. Tsai, “Global optimization of nonlinear fractional programming problems in engineering design,” *Engineering Optimization*, vol. 37, no. 4, pp. 399–409, 2005.

[127] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, “Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements,” in *2013 IEEE congress on evolutionary computation*, pp. 2337–2344, IEEE, 2013.

[128] C. A. C. Coello, “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art,” *Computer methods in applied mechanics and engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[129] F.-z. Huang, L. Wang, and Q. He, “An effective co-evolutionary differential evolution for constrained optimization,” *Applied Mathematics and computation*, vol. 186, no. 1, pp. 340–356, 2007.

[130] I. Fister, J. B. Žumer, *et al.*, “Memetic artificial bee colony algorithm for large-scale global optimization,” in *2012 IEEE Congress on evolutionary computation*, pp. 1–8, IEEE, 2012.

[131] G. D’Angelo and F. Palmieri, “Gga: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems,” *Information Sciences*, vol. 547, pp. 136–162, 2021.

[132] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, “A unified differential evolution algorithm for constrained optimization problems,” in *2017 IEEE congress on evolutionary computation (CEC)*, pp. 1231–1238, IEEE, 2017.

[133] R. Meddis, “Unified analysis of variance by ranks,” *British Journal of Mathematical and Statistical Psychology*, vol. 33, no. 1, pp. 84–98, 1980.

[134] J. Cleverley, J. Piper, and M. M. Jones, “The role of chest radiography in confirming covid-19 pneumonia,” *bmj*, vol. 370, 2020.

[135] S. H. Lee, H. I. Koo, and N. I. Cho, "Image segmentation algorithms based on the machine learning of features," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2325–2336, 2010.

[136] N. Tang, F. Zhou, Z. Gu, H. Zheng, Z. Yu, and B. Zheng, "Unsupervised pixel-wise classification for chaetoceros image segmentation," *Neurocomputing*, vol. 318, pp. 261–270, 2018.

[137] F. Breve, "Interactive image segmentation using label propagation through complex networks," *Expert Systems With Applications*, vol. 123, pp. 18–33, 2019.

[138] X. Yue and H. Zhang, "Improved hybrid bat algorithm with invasive weed and its application in image segmentation," *Arabian Journal for Science and Engineering*, vol. 44, no. 11, pp. 9221–9234, 2019.

[139] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 1, pp. 695–701, IEEE, 2005.

[140] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

[141] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[142] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[143] S. Kullback, "Information theory and statistics (peter smith, gloucester, ma)," *Information theory and statistics*, 1968.

[144] C. Li and P. K.-S. Tam, "An iterative algorithm for minimum cross entropy thresholding," *Pattern recognition letters*, vol. 19, no. 8, pp. 771–776, 1998.

[145] D. Kermany, K. Zhang, M. Goldbaum, *et al.*, "Labeled optical coherence tomography (oct) and chest x-ray images for classification," *Mendeley data*, vol. 2, no. 2, 2018.

[146] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[147] T. Rahkar Farshi and A. K Ardabili, "A hybrid firefly and particle swarm optimization algorithm applied to multilevel image thresholding," *Multimedia Systems*, vol. 27, no. 1, pp. 125–142, 2021.

[148] S. Sivanandam, S. Deepa, S. Sivanandam, and S. Deepa, *Genetic algorithms*. Springer, 2008.

[149] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[150] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in engineering software*, vol. 105, pp. 30–47, 2017.

[151] H. S. N. Alwerfali, M. Abd Elaziz, M. A. Al-Qaness, A. A. Abbasi, S. Lu, F. Liu, and L. Li, "A multilevel image thresholding based on hybrid salp swarm algorithm and fuzzy entropy," *IEEE Access*, vol. 7, pp. 181405–181422, 2019.

[152] P. Upadhyay and J. K. Chhabra, "Multilevel thresholding based image segmentation using new multistage hybrid optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1081–1098, 2021.

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

[154] W. Long, J. Jiao, X. Liang, and M. Tang, "An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 63–80, 2018.