Enhanced Capuchin Search Algorithm Using Cooperative Island Model with Application of Evolutionary Feedforward Neural Networks

Thaer Thaher*, Mohammed Awad*, Alaa Sheta†, Mohammed Aldasht‡,
**Department of Computer Systems Engineering*, *Arab American University*, Jenin, Palestine
Email: thaer.thaher@aaup.edu; mohammed.awad@aaup.edu
†*Computer Science Department*, *Southern Connecticut State University*, CT 06515, USA
Email: shetaa1@southernct.edu
‡*Department of Computer Engineering*, *Palestine Polytechnic University*, Hebron, Palestine
Email: Mohammed@ppu.edu

***Abstract*—This paper introduces an enhanced version of the Capuchin Search Algorithm (CapSA) called ECapSA. CapSA draws inspiration from the collective intelligence of Capuchin monkeys and has shown success in solving real-world problems. However, it may encounter challenges handling complex optimization tasks, such as premature convergence or being trapped in local optima. ECapSA employs a local escaping mechanism operating the abandonment limit concept to exploit potential solutions and introduce diversification trends. Additionally, the ECapSA algorithm is improved by integrating the principles of the cooperative island model, resulting in the iECapSA. This modification enables better management of population diversity and a more optimal balance between exploration and exploitation. The efficiency of iECapSA is validated through a series of experiments, including the IEEE-CEC2014 benchmark functions and training the feedforward neural network (FNN) on seven biomedical datasets. The performance of iECapSA is compared to other metaheuristic techniques, namely differential evolution (DE), sine cosine algorithm (SCA), and whale optimization algorithm (WOA). The results of the comparative study demonstrate that iECapSA is a strong contender and surpasses other training algorithms in most datasets, particularly in terms of its ability to avoid local optima and its improved convergence speed.**

***Index Terms*—Capuchin search algorithm, island model, population diversity, training neural networks.**

I. Introduction

The optimization field deals with determining the best possible values for a group of decision variables that either maximize or minimize an objective function [1]. To discover the best possible solution for the optimization problem, a robust optimization algorithm is necessary to explore unvisited regions and take advantage of previously explored areas of the search space [2]. Evolutionary Algorithms (EAs) are commonly used optimization algorithms that can solve various optimization problems [3]. EAs start with a population of individuals and improve their fitness over generations using intelligent operators and problem-specific knowledge until an optimal solution is achieved. EAs can be classified into different categories based on their natural inspirations, such as swarm-based, evolutionary-based, physical-based, chemical-based, human-based, and mathematical-based [4].

Swarm-based EAs draw inspiration from animal behavior, particularly in the areas of prey hunting, collaborative food-finding, living style, and swarm leadership. These algorithms typically consist of two groups: leaders and followers. The followers are attracted to the leader's group to improve their search directions. Particle swarm optimization (PSO) [5] and ant colony optimization (ACO) [6] are examples of base swarm intelligence algorithms, but numerous other algorithms have been proposed recently, including the Grey Wolf Optimizer (GWO), Moth-Flame Optimization Algorithm (MFO), Flower Pollination Algorithm (FPA), Krill Herd Algorithm (KHA), Squirrel Search Algorithm (SSA), and Crow Search Algorithm (CSA) [7].

The Capuchin search algorithm (CapSA) is a recent SI optimization technique proposed by Braik et al. in 2021. It is inspired by the foraging behavior of capuchin monkeys, who efficiently search for food by moving between different trees and branches [8]. CapSA has shown promising results in solving a variety of optimization problems, including photovoltaic parameter extraction [9], prediction of wind power [10], feature selection [10], task scheduling in the cloud computing [11], economic load dispatch problem [12], data aggregation routing protocol in the IoT [13], image segmentation [14], and parameters optimization of plastic injection molding [15]. This can be attributed to CapSA's impressive characteristics, such as simplicity, flexibility, and soundness. CapSA has a good balance between exploration and exploitation by combining local and global search strategies, which allows it to converge to high-quality solutions efficiently. Moreover, the algorithm has a simple and intuitive design, making it easy to implement and understand. Additionally, the algorithm incorporates a dynamic balance factor that helps to maintain the diversity of the solutions during the search process. while CapSA has shown competitive performance on some challenging problems, it has limitations like any other optimization algorithm. The performance of the CapSA is significantly influenced by its parameter setup, and identifying suitable parameter values for each problem can be challenging. Furthermore, the CapSA may require greater diversity in its population to prevent premature convergence and suboptimal outcomes. Specifically, the CapSA's propensity for getting stuck in local optima presents a potential limitation that necessitates further investigation and testing, especially when confronted with more complex real-world problems [12], [16].

In the past, metaheuristic search algorithms have demonstrated their usefulness in training neural networks due to their capability to handle complex, non-convex optimization problems, which traditional optimization methods find difficult to address. When training neural networks, the objective is to determine the optimal values for numerous weights and biases, resulting in a high-dimensional and non-convex optimization problem. Metaheuristic search algorithms such as genetic algorithm (GA), PSO, and simulated annealing (SA) have successfully identified acceptable solutions for such problems, even without explicit knowledge of the problem's landscape. These algorithms can explore the search space effectively and locate favorable regions to concentrate on, resulting in faster convergence and improved performance for the neural network. Therefore, employing ECapSA can enhance the neural network training process and facilitate better accuracy and generalization ability [17].

The strength and power of the learning mechanism play a critical role in the performance of FNN. Several previous studies have employed gradient-based backpropagation methods. However, this algorithm is associated with two primary issues, namely, slow convergence rate and local minima trap [18]. As a result, metaheuristic-based approaches are being developed to address these concerns associated with the gradient-based algorithm. Many swarm intelligence algorithms are utilized in the literature to handle the training of FNNs, for example, artificial bee colony (ABC), PSO, teaching-learning-based optimization (TLBO), whale optimization algorithm (WOA), salp swarm algorithm (SSA), and others. A review of metaheuristic design for FNN can be found in [19], while a thorough analysis comparing the effectiveness of various metaheuristic algorithms for training FNN can be found in [20].

A. *Motivations and contributions*

While training-based metaheuristic algorithms are effective in accelerating the convergence rate and avoiding local minima, which is a problem with gradient-descent algorithms, they cannot guarantee an exact solution in the optimization domain. Additionally, the No Free Lunch theorem (NFL) suggests no universal optimization technique can outperform all others for all optimization problems. Effective metaheuristic-based methodologies are still being researched, with this paper focusing on overcoming the main drawbacks of the primary CapSA algorithm by proposing a modified version and adapting the cooperative island model concept as a training algorithm to enhance the performance of FNNs. The main contributions of this paper are summarized as follows:

- We introduce a local escaping mechanism based on the abandonment limit concept to enhance CapSA, resulting in a new and improved version called ECapSA. By incorporating this strategy, the algorithm can better exploit potential solutions and introduce diversification trends, ultimately leading to improved convergence performance.
- The fundamentals of the island model are incorporated into the structure of ECapSA, referred to as iECapSA, to manage diversity in the population, and prevent premature convergence.
- iECapSA introduces a novel FNN trainer for use in the biomedical field. When compared to other state-of-the-art algorithms, iECapSA produces high-quality results.

The article is structured as follows: Section II provides an overview of CapSA and FNNs, highlighting their key features. Section III-B provides an in-depth explanation of the proposed iECapSA algorithm. The experiments conducted and the results obtained are described in Section IV. Lastly, Section V summarizes the study's findings.

II. Research background

This section provides an overview of the methods used to develop the proposed optimization model to create a self-exploratory paper. It begins by explaining the basic CapSA in Section II-A, and then delves into the principles of feedforward neural network in Section II-B.

A. *Overview of Capuchin Search Algorithm (CapSA)*

A novel metaheuristic called CapSA was developed by studying the natural behavior and daily routines of capuchin monkeys as they foraged for food in the wild. According to Braik et al., [8], the most fascinating fact regarding the social behavior of capuchin monkeys is essential that they employ three excellent maneuvers to move about when foraging on trees, riverbanks, and ground, namely jumping, swinging, and climbing. These facts form the basis of CSA's fundamental assumptions. The CapSA algorithm, like other swarm intelligence-based algorithms, may be thought of as a population-based algorithm that starts by randomly initializing a preset number of individuals (i.e., capuchins). Each capuchin represents a possible solution to the problem being addressed. The mathematical representation of CapSA's evolutionary process is as follows: Several N capuchins are assumed to be distributed over a d-dimensional search space. N is often referred to as population size. A possible solution to the problem of interest may be determined by the location of each capuchin individual i at a specific moment (t), which can be described as a vector:

$$X_i(t) = \left[x_i^1, x_i^2, x_i^3 \dots x_i^d\right]$$

Each individual is also characterized by its velocity:

$$V_i(t) = \left[v_i^1, v_i^2, v_i^3 \dots v_i^d\right]$$

where $(i = 1, 2, , ..., N)$, $(t = 1, 2, ...T)$, T is the maximum number of iterations, d is the number of variables of a test problem, and x_d^i denotes the dth dimension of the ith capuchin. The whole population of capuchins X and the corresponding velocities V are initially positioned randomly in the d-dimensional search space.

Capuchin swarms often consist of two categories of individuals: (1) leaders, also known as alphas, who are responsible for finding new food sources, and (2) followers, who are in charge of updating their positions by following the group's leaders.

1) Leaders updating rules: The community's leaders employ five distinct mobility strategies to locate food during the evolutionary process. A random number named ϵ is generated to determine operation selection as follows:

- **Jumping on trees** ($i < N/2; 0.1 < \epsilon \leq 0.15$): In this situation, alpha capuchins can be positioned using the following updating rule:

$$X_i(t+1) = gbest + \Big(\frac{P_{bf}(v_i)^2 sin(2\theta)}{g}\Big) \qquad (1)$$

  where the variable $gbest$ denotes the current optimal position of the food source, while ϵ refers to a randomly generated number in the range of 0 to 1. P_{bf} represents the probability of the capuchin monkey's tail providing balance during the jumping process. The velocity of the ith capuchin, denoted as v_i, is computed using Equation (2), with g being the gravitational acceleration constant set to 9.81. The jumping angle, denoted as θ, is calculated as $1.5 times r$, where r is a uniformly distributed random number between 0 and 1.

$$\begin{aligned} V_i(t+1) &= \rho v_i(t) + a_1\Big(pbest_i - x_i(t)\Big)r_1 \\ &+ a_2\Big(gbest - x_i(t)\Big)r_2 \end{aligned} \qquad (2)$$

  where $pbest$ refers to the best position so far of the ith capuchin. The factors a_1 and a_2 govern the influence of the individual best position ($pbest$) and the global best position ($gbest$) on the capuchin's velocity. The random variables r_1 and r_2 are uniformly distributed in the interval [0, 1]. The inertia coefficient ρ determines how much the previous velocity affects the current motion, and in this study, it is decreased during iterations using Equation (3) to regulate the search for either local or global solutions.

$$\rho = w_{max} - (w_{max} - w_{min})(\frac{t}{T})^2 \qquad (3)$$

  where w_{max} and w_{min} take values of 0.8 and 0.1, respectively.

- **Jumping on the ground** ($i < N/2; 0.15 < \epsilon \leq 0.3$): Capuchins employ this behavior to travel long distances, especially when food is hard to come by on the trees. The new position of the leader and following capuchins in this instance may be determined as follows:

$$X_i(t+1) = gbest + \Big(\frac{P_{ef}P_{bf}(v_i)^2 sin(2\theta)}{g}\Big) \qquad (4)$$

  where P_{ef} stands for the elasticity probability of the capuchin movement on the ground.

- **Normal walking on the ground** ($i < N/2; 0.3 < \epsilon \leq 0.9$): In this case, the leaders' position can be updated as follows:

$$X_i(t+1) = x_i(t) + v_i(t+1) \qquad (5)$$

- **Swinging on the trees** ($i < N/2; 0.9 < \epsilon \leq 0.95$): While looking for food on tree branches, certain alpha capuchins and other accompanying capuchins may employ local search. The following rule was used to simulate this behavior:

$$X_i(t+1) = gbest + P_{bf} \times sin(2\theta) \qquad (6)$$

- **Climbing trees** ($i < N/2; 0.95 < \epsilon \leq 1.0$): Certain alpha capuchins and other following capuchins may repeatedly ascend and descend trees and their branches in a manner akin to local search. Specifically, the following rule can be used to update the capuchins' location:

$$X_i(t+1) = gbest + P_{bf}\Big(v_i(t+1) - v_i(t)\Big) \qquad (7)$$

- **Random migration of the capuchines** ($i < N/2; \epsilon < 0.1$): Capuchin monkeys engage in random migration during food foraging, wherein they search for food in various directions to more efficiently explore their surroundings in search of better food sources. The process of random migration is modeled using Eq. (8)

$$X_i(t+1) = \tau \times \Big[LB + r \times (UB - LB)\Big] \qquad (8)$$

  where LB and UB represent the lower and upper bounds of the decision variables. Capuchin monkeys have a 0.1 probability of engaging in a random search. This strategy enhances CapSA's capacity to explore globally and reduces the likelihood of getting stuck in local optima. In CapSA, an exponential function with a lifetime parameter (τ) was introduced to balance exploration and exploitation during global and local search processes. This function is represented by Eq.

$$\tau = 2e^{-11(\frac{t}{T})^2} \qquad (9)$$

In summary, according to Eqs. (1) through (9), the capuchin monkeys adjust their positions based on the presence of food. This behavior is especially noticeable when $r > 0.1$. However, if $r \leq 0.1$, the capuchin monkeys tend to change their positions randomly to explore different areas for food. In such instances, the parameter τ can expand the exploration space for searching.

2) Followers updating rule: The positions of the followers (i.e. $N/2 \leq i \leq N$) are updated according to the following formula:

$$x_i(t+1) = \frac{1}{2}\big(x_i(t) + x_{i-1}(t)\big) \qquad (10)$$

where $x_i(t)$ and $x_{i-1}(t)$ are the positions of the i-th follower and the $(i-1)$-th follower in in the previous generation. The original paper describes the comprehensive steps and the complete mathematical model utilized to derive this formula.

B. Feedforward Neural Networks

A feedforward neural network, known as FNN, is an artificial neural network where data flows in a unidirectional manner, moving from the input layer through the hidden layers to the output layer, without any feedback connections [21]. In this architecture, the input layer accepts the input data, the hidden layer(s) process the input data using a set of weights and activation functions, and the output layer generates the final output. Figure 1 provides an example of an FNN that has three input features, one hidden layer, and one output layer.

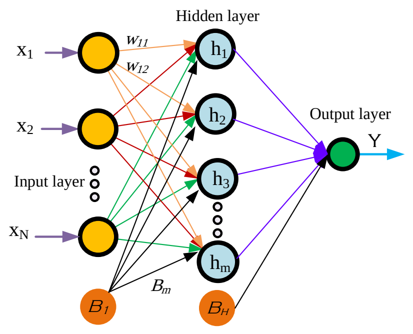


Fig. 1: Simple FNN architecture with one hidden layer

The mathematical model of the FNN relies on three primary components: input features, biases, and weights. The input layer receives a vector of features (i.e., input variables), while each neuron in the other layers performs a summation function and an activation function. The summation function calculates the weighted sum of inputs by multiplying them with their corresponding weights, adding a bias term, and applying an activation function, as shown in Equation (11).

$$S_j = \sum_{i=1}^{n} w_{ij} \times X_i + B_j \qquad j = 1, 2, ...m \tag{11}$$

where m represents the total number of hidden nodes. Meanwhile, n stands for the total number of input nodes. Each connection between the ith input node X_i and the jth hidden node has a connection weight w_{ij}, and each hidden neuron j has a bias term β_j.

Once the aggregation function defined in Equation (11) is computed, an activation function is applied to activate the neurons' output. FNN networks can utilize several types of activation functions. In this research, the sigmoid function is used, which has been frequently applied in previous studies involving FNN networks [17], [22]. This function is used to propagate the weighted output of the hidden layer to the subsequent layer. Equation (12) is used to determine the output of node j in the hidden layer.

$$h_j = \frac{1}{1 + e^{-S_j}} \qquad j = 1, 2, ...m \tag{12}$$

where h_j represents the sigmoid activation function that is applied to the jth node in the middle layer, while S_j refers to the summation obtained from Equation (11). Once the output for each neuron in the middle layer is computed, the next step is to determine the output of the FNN network. This can be done by applying Equation (13).

$$\hat{y} = sigmoid\left(\sum_{j=1}^{m} w_j \times h_j + \beta\right) \tag{13}$$

After constructing the neural network, the weights linked with the network are adapted to approximate the desired outcomes. To accomplish this, a training algorithm is utilized to modify the weights in an iterative manner until a certain error criterion is met.

III. The proposed ECapSA algorithm

Since its inception, the CapSA algorithm has been widely applied in various research areas and is highly effective. However, despite its numerous positive aspects, the algorithm still faces various issues [16]. Like most metaheuristics, CapSA is susceptible to the lack of diversity in its population. To be specific, CapSA has a tendency to become trapped in local optima, which could limit its effectiveness. Consequently, to improve the performance of CapSA, it is necessary to incorporate additional operators that emphasize exploitation and provide a better balance between diversification and intensification. In the subsequent sections, we present proposed enhancements to the CapSA algorithm that aim to achieve these goals.

A. Boosting CapSA with local escaping mechanism (ECapSA)

The basic CapSA mathematical model indicates that during the evolution process, information from the local best ($pbest$) of each individual in the population is utilized in exploration procedures. This article proposes an improved version of the CapSA algorithm called ECapSA, which leverages the benefits of the local best optimum. ECapSA integrates two distinct update processes for each individual in the population, depending on whether an abandonment limit criterion is met.

- If the ith best local optimum ($pbest_i$) experiences an improvement, the first update process follows the original CapSA. However, if no improvement is observed in the best local position ($pbest_i$) after K iterations, the update process for the corresponding individual x_i is discarded.
- Instead, a second update process is employed, which applies a perturbation to the local optimum using Eq. (14).

$$X_i(t+1) = pbest_i + r\big(x_{rand} - x_i(t)\big) \tag{14}$$

where $pbest_i$ denotes the local best position of the ith individual, x_{rand} indicates a randomly chosen position from a set of N solutions, and r signifies a random number uniformly generated within [0, 1].

Algorithm 1 outlines the various steps involved in the proposed ECapSA. This process ensures that if there is no progress in the local best solution after a set number of iterations, the leaders/followers update mechanism is replaced by another mechanism that is based on a differential evolution perturbation of the ith best local optimum [23]. This mechanism allows the corresponding capuchin to explore the neighborhood of its local best so far. This is achieved by introducing a step size that is determined by the difference between the randomly selected position and the capuchin's current position to detect a better solution. This process enables a better exploitation process around the existing best local optima and thus improves the quality of the population.

Algorithm 1 Pseudo-code of the enhanced CapSA (ECapSA)

```
1: Initialize the adjustable parameters of the basic CapSA
2: Define the parameter for the abandonment limit ($k$)
3: define $count_i = 0$ (i=1,2,...,N)
4: Generate the initial positions of the capuchines $x_i (i = 1, 2, \ldots, N)$ randomly.
5: Calculate the fitness of each individual capuchin
6: Initialize the velocity $v_i$ and memory $pbest_i$ of capuchins
7: $gbest$ = the optimal solution so far
8: while ($t < T$) do
9:     Update $\tau$ using Eq. (9)
10:    for $i = 1$ to $N$ do
11:        if ($count_i \leq K$) then
               Update $x_i$ following the rules of the original
               CapSA through Eqs. (1) to (10).
12:        else
13:            Update $x_i$ using Eq. (14)
14:            $count_i$ = 0
15:        end if
           Adjust $X_i$ through the limits set for the variable's
           upper and lower boundaries.
16:    end for
17:    Evaluate the fitness of each new position $f(x_i)$
18:    for $i = 1$ to $N$ do
19:        if $f(x_i) < f(pbest_i)$ then
20:            $pbest_i = x_i$
21:            $f(pbest_i) = f(x_i)$
22:        else
23:            $count_i = count_i + 1$
24:        end if
25:        if $f(x_i) < f(gbest)$ then
26:            $gbest = x_i$
27:            $f(gbest) = f(x_i)$
28:        end if
29:    end for
30:    $t = t + 1$
31: end while
32: Return $gbest$
```

B. island-based enhanced CapSA (iECapSA)

Metaheuristic algorithms like CapSA and ECapSA use a population-based approach to find optimal or near-optimal solutions. They generate a set of solutions, evaluate their quality, and create a new population for the next iteration. However, this approach can lead to premature convergence, where the algorithm gets stuck in a local optimum and doesn't explore the entire search space. To avoid this, the island model can be used to introduce diversity in the population.

An island model is a popular approach in distributed computing, where the population of candidate solutions is partitioned into multiple subpopulations or "islands." that evolve independently [24]. Each island has its instance of algorithm, which can use different parameters or operators. The islands cooperate by exchanging solutions periodically to propagate good solutions and promote diversity. Consequently, the island model is motivated by the need to balance exploration and exploitation in metaheuristic optimization [25].

The island model typically involves the periodic exchange of candidate solutions between the islands, which is called migration. The migration parameters refer to the settings that control the transfer of solutions between subpopulations during the optimization process. These parameters include the frequency of migration (M_f), the number of individuals to be migrated (M_r), the selection method for choosing individuals for migration (M_s), and the topology of the islands (M_t) [26]. The frequency of migration is the number of iterations after which migration occurs. It can be fixed or adaptive based on the progress of the search process. The number of individuals to be migrated can be a fixed number or a percentage of the island population size. The selection method for choosing individuals for migration can be based on different criteria, such as the fitness value, the diversity of the candidate solutions, or a combination of both. The topology of the islands can be defined as a ring, a fully connected graph, or any other graph structure that allows the exchange of candidate solutions between the islands. The migration process can be done in different ways, such as the sending of the best individuals or randomly selected individuals, or by using crossover operators to create new candidate solutions in the destination island.

In this study, the island model is incorporated into the structure of ECapSA, referred to as iECapSA, to manage diversity and enhance the efficiency of searching the domain. Specifically, the proposed iECapSA follows the steps outlined below:

- **Step 1: Initialize Problem and Adjustable Parameters** In this stage, the necessary parameters are initialized, including the problem parameters, ECapSA parameters, and migration parameters.
- **Step 2: Generate the Initial Population** In this step, the initial population of capuchins, represented as X = (x_1, x_2, x_3, ... x_N), is randomly positioned in the d-dimensional search space as potential solutions. Each candidate solution is generated using Eq. (15).

$$\vec{X}_i = \vec{X}_L + r(\vec{X}_U - \vec{X}_L) \tag{15}$$

  where $\vec{X}_L$ and $\vec{X}_U$ are the lower and upper bound for the problem dimensions, r is a random number inside [0,1].
- **Step 3: Splitting the initial population into islands** As Step 3 of the algorithm, the initial population is partitioned into a specified number of islands denoted as s, with each island comprising of $I_s = N/s$ solutions, where N is the total population size.
- **Step 4: Running the Optimization Process**

In this step, the optimization process is initiated by running the cooperative algorithms concurrently. Each island has its own population and search mechanism utilizing ECapSA. It's important to note that the same search algorithm with the same settings is embedded in each island, creating a homogeneous island model. The evolution process occurs asynchronously, based on the generalized island model [27].

- **Step 5: Migration Process**
  After a certain number of iterations are determined by the migration frequency parameter (M_f), a migration process is initiated. This involves generating a random communication topology, such as a bidirectional ring, to connect the islands. A percentage of the individuals from each island, determined by the migration rate parameter (M_r), is then transferred to the connected islands. The best-worst policy is used for the swapping process between every two neighboring islands.
- **Step 6: Stop criterion**:
  The optimization process continues by repeating steps 4-6 until the stopping criterion is met.
- **Step 7: Return the best solution**:
  Once the stop condition is met, the algorithm returns the best solution found among all the islands.

C. Proposed iECapSA-FNN model

To adapt SI algorithms for optimization problems, two key steps are necessary: defining the problem representation and formulating the objective function. In this study, the primary aim is to find the optimal values for weights and biases of a single hidden layer FNN that would yield the lowest prediction error. As such, a vector of real values was employed to encode the solution, as described by Eq. (16).

$$\vec{X} = \{\vec{W}, \vec{\beta}\} \tag{16}$$

where the weights and biases are denoted as $\vec{W}$ and $\vec{\beta}$ respectively, and their values are assumed to be within the range of [-1, 1] [22]. Although there is no established method in the literature for determining the optimal number of neurons in the hidden layer, in this study, the approach proposed in [17], [22] is utilized. This technique sets the number of hidden neurons (H) as (2 × F + 1), where F represents the number of features in the dataset. Consequently, the dimension of each solution (i.e., D) is calculated as shown in Eq. (17).

$$D = (F \times H) + (2 \times H) + 1 \tag{17}$$

Once the solution representation in FNN has been established, the subsequent step is to formulate an objective function to assess the quality of the produced solutions. In this study, we employed the Mean Squared Error (MSE) as our chosen metric for evaluating FNNs, as it is a widely used technique for this purpose [22]. The assessment of solutions involved inputting the generated weights and biases into the FNN and then computing the MSE, which is represented in equation (18).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{18}$$

where y and $\hat{y}$ represent the actual and estimated values, respectively. n is the number of training samples.

IV. Experimental results and discussion

Within this section, we evaluate the capacity of the developed iECapSA to address a range of diverse optimization problems, such as CEC2014, and training of FNNs. As a result, the paper features three primary experiments: (1) a comparison experiment that seeks to determine whether the proposed strategy has a positive impact on CapSA, (2) a performance validation experiment for the proposed iECapSA variant, and (3) an experiment on Training FNN based on iECapSA. The first two experiments were conducted during 5 selected mathematical functions from the IEEE CEC2014 test suit, while the training of the FNN experiment relied on seven biomedical datasets.

A. Test problems

1) Mathematical benchmark functions CEC2014: The literature offers various mathematical optimization problems to assess the effectiveness of optimization algorithms, including their robustness, convergence rate, and overall performance. To evaluate the proposed methods, five problems with different characteristics were selected from the IEEE CEC2014 suit. These functions are classified into four categories: unimodal function (F1), multimodal function (F10 and F14), hybrid function (F22), and composition function (F26). Table I summarizes the main characteristics of these functions.

TABLE I: The characteristics of CEC2014 benchmark functions (U: Unimodal, M: Multimodal, H: Hybrid, C: Composition)

Function	Description	Category	F_{min}
F1	Rotated High Conditioned Elliptic Function	U	100
F10	Shifted Schwefel's Function	M	1000
F14	Shifted and Rotated HGBat Function	M	1400
F22	Hybrid Function 6 (N=5)	H	2200
F26	Composition Function 4 (N=5)	C	2600
	D = 30 & Range $[-100, 100]^D$		

2) Binary classification datasets: To evaluate and compare the effectiveness of the proposed iECapSA approach for training FNNs, seven real biomedical datasets were chosen from the UCI Machine Learning Repository [28], all of which involve binary classification. Table II provides a detailed description of these datasets.

B. Experimental setup and settings

All experiments were conducted on the same computing environment, using a machine equipped with an Intel(R) Core(TM) i7-1165G7 CPU running at 2.80 GHz (with 8 CPUs) and 16 GB of RAM, operating on Ubuntu 20.04 LTS.

TABLE II: A summary of each binary classification dataset and its corresponding FNN structure

Dataset	#Features	#samples	Hidden Layer	FNN structure
Blood	4	748	9	4-9-2
BreastCancer	8	699	17	8-17-2
Diabetes	8	768	17	8-17-2
diagnosis_II	6	120	13	6-13-2
Liver	6	345	13	6-13-2
Parkinsons	22	195	45	22-45-2
Vertebral	6	310	13	6-13-2

In the FNN training experiment, the datasets were divided into training and testing sets, with 66% of the data used for training and 34% for testing. To ensure reliable results, each experiment was repeated 30 times, with each run consisting of 250 iterations. The standard parameter settings for iECapSA, DE, WOA, and SCA as recommended in the literature, are summarized in Table III and were utilized in our experiments.

TABLE III: Parameter settings of iECapSA and other algorithms

Common parameters		
Population size N		50
Maximum No. of iterations		1000 (CEC2014), 250 (FNN)
No. of runs		30
significance level α (Friedman test)		0.05
Internal parameters		
Algorithm	parameter	value
iECapSA	a_1, a_2	1.25, 1.5
	P_{ef}, P_{bf}	11 , 0.7
	w_{min} , w_{max}	0.1, 0.8
	abandonment limit K	10
	Number of islands	4
	Migration frequency M_f	0.2 $\times$ T
	Migration rate M_r	0.3 $\times N$
	Migration policy	best-worst
	Communication topology	random ring
WOA	convergence constant a	decreased linearly [2 0]
	Spiral factor b	1
DE	Mutation, Crossover	0.5 , 0.7
SCA	$r1$	decreased linearly [2 0]
	$r2$	random values inside [0 2π]
	$r3$	random values inside [0 2]
	$r4$	random values inside [0 1]

C. Evaluation measures

The effectiveness of the proposed method is evaluated using various performance measures, including the mean of fitness values over 30 independent runs. In addition, the accuracy measure is used for the FNN. Moreover, population diversity is an essential performance measure for metaheuristics. A diverse population indicates that the algorithm is effectively exploring the search space and discovering different regions of good solutions, while a low diversity population indicates that the algorithm may be stuck in a local optimum. In this study, an effective measure based on the idea of the moment of inertia [29] is utilized to quantify population diversity.

D. Experimental series 1: CEC2014

The first scenario involves conducting experiments on five challenging CEC2014 benchmark test functions to analyze the effects of the proposed search strategies on the primary CapSA. To this end, we evaluated two developed variants, namely ECapSA and iECapSA, in terms of their solution quality, convergence trends, and diversity curves. The results of 30 independent runs are presented in Table IV, where the best-performing solution is highlighted in **bold**. The findings show that the proposed ECapSA method outperforms the standard CapSA in 80% of the test functions (F1-F4). Furthermore, the island-based ECapSA (iECapSA) consistently outperforms all other methods in all test cases. These results suggest that integrating the new search strategy into CapSA and combining it with the principles of the island model can enhance its exploitation potential and improve its exploration capacity to search the entire solution space.

TABLE IV: Comparison of CapSA variants on 30-dimensional IEEE CEC2014

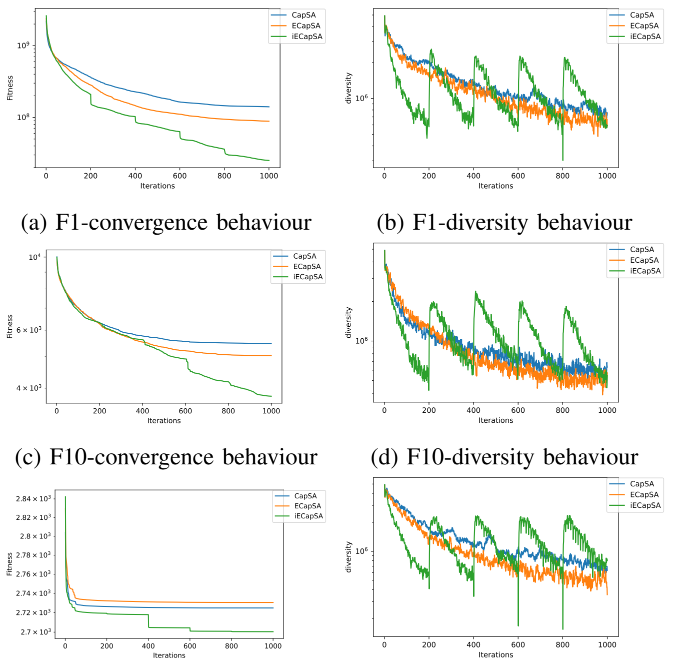
Function	CapSA	ECapSA	iECapSA
F1	1.41E+08	8.86E+07	**2.51E+07**
F10	5.46E+03	5.02E+03	**3.78E+03**
F14	1.43E+03	1.42E+03	**1.40E+03**
F22	3.14E+03	2.86E+03	**2.61E+03**
F26	2.72E+03	2.73E+03	**2.70E+03**
Mean Rank	2.80	2.20	**1.00**

In the visualization presented in Figure 2, the convergence and population diversity curves of $CapSA$, $ECapSA$, and $iECapSA$ are depicted side by side for selected test functions. These figures enable the assessment of the impact of population diversity on the optimization process. In specific, the convergence curve illustrates how the fitness value evolves over the course of iterations. The diversity curve shows the variation or diversity of solutions within the population over the course of iterations. The convergence curves clearly depict the significant improvements in the convergence rate of the proposed methods. These results offer additional evidence that the suggested techniques perform exceptionally well in accelerating the convergence rate.

Upon examining Figure 2 with regard to diversity analysis, it is evident that all CapSA variants exhibit a significant level of diversity at the start of the optimization process, indicating an extensive exploration of the search space. However, as the number of iterations increases, the population's diversity gradually decreases. The figure shows that $CapSA$ and $ECapSA$ have smoother diversity responses, suggesting more exploitative behavior for the proposed $ECapSA$ algorithm. In contrast, the $iECapSA$ algorithm displays highly oscillating behavior, which reflects its superior exploration-exploitation capabilities. These results confirm that the migration policy in the iECapSA model, which involves exchanging information among different populations, can introduce additional variation in the diversity of candidate solutions and result in more oscillations in the diversity curves.

E. Experimental series 2: Training of FNN

In this section, the performance of the proposed algorithms, ECapSA and iECapSA, is evaluated on the training of FNN by conducting experiments and comparing them with the original CapSA and other well-established algorithms namely DE, WOA, and SCA. The reason for selecting these algorithms is that they encompass a variety of classes of metaheuristics,



(a) F1-convergence behaviour (b) F1-diversity behaviour

(c) F10-convergence behaviour (d) F10-diversity behaviour

(e) F26-convergence behaviour (f) F26-diversity behaviour

Fig. 2: The convergence and diversity plots of CapSA variants on selected CEC2014 functions

in terms of both inspiration and mathematical formulation. Additionally, these algorithms include a mix of novel techniques such as WOA and SCA, as well as some of the most popular optimizers used in this field, such as DE. The effectiveness of the proposed algorithm is measured using a fitness function (MSE) and classification quality metric (accuracy).

TABLE V: Testing accuracy results

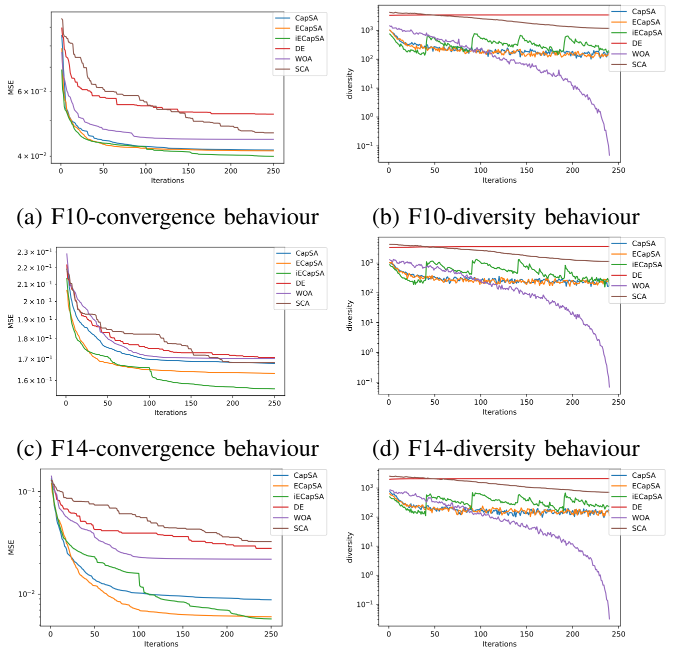
Dataset	CapSA	ECapSA	iECapSA	DE	WOA	SCA
Blood	0.7525	0.7510	0.7522	0.7533	0.7553	**0.7561**
BreastCancer	0.9689	0.9693	**0.9723**	0.9571	0.9702	0.9643
Diabetes	0.7294	0.7389	**0.7519**	0.7183	0.7179	0.7385
diagnosis_II	**1.0000**	**1.0000**	**1.0000**	0.9976	0.9951	**1.0000**
Liver	0.6788	0.6966	**0.7415**	0.6983	0.6542	0.6958
Parkinsons	**0.8343**	0.8299	0.8328	0.7478	0.7776	0.7716
Vertebral	0.8236	0.8387	**0.8575**	0.8057	0.8075	0.8104
Mean Rank	3.36	3.07	1.93	4.71	4.43	3.50

Table V presents a summary of the mean classification accuracy results. The proposed iECapSA algorithm surpasses all other optimizers on four datasets (BreastCancer, Diabetes, Liver, and Vertebral), with average accuracy values of 0.9723, 0.7519, 0.7415, and 0.8575, respectively. Additionally, iECapSA exhibits an average accuracy of 1.00 for Diagnosis-II, which is consistent with the results of CapSA, ECapSA, and SCA. These results highlight the algorithm's ability to outperform other comparative techniques in navigating the problem search space, while also being capable of escaping from local optima due to its diversification ability.

Table VI summarizes the average MSE obtained by the comparative methods for all datasets. The results demonstrate that the iECapSA provides extremely competitive results and reaches the best outcomes for all cases. $iECapSA$ ranks first, followed by $ECapSA$, $CapSA$, SCA, DE, and WOA. In summary, $iEcapSA$ shows remarkable optimization results for the optimization of FNN. The algorithm's capacity to balance between exploitation and exploration, allows it to explore new areas of the search space and provide superior solutions. Figure 3 illustrates the convergence patterns of the comparative algorithms. The figure makes it evident that the $iECapSA$ algorithm exhibits notably faster convergence rates than the conventional $CapSA$ and other techniques.

TABLE VI: MSE Results

Dataset	CapSA	ECapSA	iECapSA	DE	WOA	SCA
Blood	1.56E-01	1.56E-01	**1.54E-01**	1.57E-01	1.57E-01	1.57E-01
BreastCancer	4.16E-02	4.14E-02	**4.00E-02**	5.21E-02	4.45E-02	4.63E-02
Diabetes	1.68E-01	1.63E-01	**1.56E-01**	1.71E-01	1.70E-01	1.68E-01
diagnosis_II	8.82E-03	6.03E-03	**5.74E-03**	2.80E-02	2.19E-02	3.27E-02
Liver	2.16E-01	2.11E-01	**2.04E-01**	2.14E-01	2.19E-01	2.16E-01
Parkinsons	1.27E-01	1.20E-01	**1.15E-01**	2.10E-01	1.63E-01	1.59E-01
Vertebral	1.48E-01	1.48E-01	**1.38E-01**	1.50E-01	1.51E-01	1.51E-01
Mean Rank	3.00	2.14	1.00	5.00	5.14	4.71



(a) F10-convergence behaviour (b) F10-diversity behaviour

(c) F14-convergence behaviour (d) F14-diversity behaviour

(e) F26-convergence behaviour (f) F26-diversity behaviour

Fig. 3: The convergence and diversity plots of the proposed CapSA variants against DE, WOA, and SCA on different classification datasets

By examining the diversity curves depicted in Figure 3, it can be observed that the $iECapSA$, which displays an oscillatory behavior between high and medium levels of diversity, yields better solution quality in most cases. This behavior indicates that $iECapSA$ effectively balances exploration and exploitation by maintaining a moderate level of diversity during the optimization process. In contrast, algorithms like DE and WOA, with either very high or very low diversity levels, do not yield the best solution quality. These results suggest that achieving an optimal balance between exploration and exploitation through an appropriate level of diversity is necessary for obtaining the best solutions.

V. Conclusion and future works

This paper presented an enhanced version of the Capuchin search algorithm ($CapSA$) named $iECapSA$. Our algorithm addresses the limitations of the standard $CapSA$ by utilizing various strategies, such as the local escaping operator and the cooperative island model's structure. The local escaping mechanism leverages the abandonment limit concept to enhance exploitation ability, while the island model helps control population diversity. We evaluate the performance of $iECapSA$ through experiments that include the CEC2014 benchmark and FNN training mechanism. Our evaluation metrics, such as fitness value, accuracy, convergence performance, and diversity, show the distinct advantages of $iECapSA$ over the original $CapSA$ and other competitive methods.

Future research will focus on improving computation time and exploring practical applications of $iECapSA$, such as software reliability optimization, image segmentation, and feature selection.

References

[1] I. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operational Research*, vol. 63, pp. 513–628, 10 1996.

[2] W. Li, G.-G. Wang, and A. Gandomi, "A survey of learning-based intelligent optimization algorithms," *Archives of Computational Methods in Engineering*, vol. 28, p. 3781–3799, 02 2021.

[3] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[4] F. Fausto, A. Reyna Orta, E. Cuevas, A. Andrade, and M. Cisneros, "From ants to whales: metaheuristics for all tastes," *Artificial Intelligence Review*, vol. 53, p. 753–810, 01 2020.

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

[6] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2, 1999, pp. 1470–1477.

[7] R. R. Mostafa, M. A. Gaheen, M. Abd ElAziz, M. A. Al-Betar, and A. A. Ewees, "An improved gorilla troops optimizer for global optimization problems and feature selection," *Knowledge-Based Systems*, vol. 269, p. 110462, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123002125

[8] M. Braik, A. Sheta, and H. Al-Hiary, "A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm," *Neural Computing and Applications*, vol. 33, no. 7, p. 2515–2547, 09 2021.

[9] H. H. Ali, A. Fathy, M. Al-Dhaifallah, A. Y. Abdelaziz, and M. Ebeed, "An efficient capuchin search algorithm for extracting the parameters of different pv cells/modules," *Frontiers in Energy Research*, vol. 10, 2022.

[10] M. A. Al-qaness, A. A. Ewees, H. Fan, L. Abualigah, A. H. Elsheikh, and M. Abd Elaziz, "Wind power prediction using random vector functional link network with capuchin search algorithm," *Ain Shams Engineering Journal*, vol. 14, no. 9, p. 102095, 2022.

[11] S. Ramu, R. Ranganathan, and R. Ramamoorthy, "Capuchin search algorithm based task scheduling in cloud computing environment," *Yanbu Journal of Engineering and Science*, vol. 19, no. 1, pp. 18–29, 3 2022.

[12] M. S. Braik, M. Awadallah, M. A. Al-Betar, and A. Hammouri, "A hybrid capuchin search algorithm with gradient search algorithm for economic dispatch problem," 2022. [Online]. Available: https://doi.org/10.21203/rs.3.rs-1761466/v1

[13] M. Mohseni, F. Amirghafouri, and B. Pourghebleh, "Cedar: A cluster-based energy-aware data aggregation routing protocol in the internet of things using capuchin search algorithm and fuzzy logic," *Peer-to-Peer Networking and Applications*, vol. 16, p. 189–209, 10 2022.

[14] S. Li, Z. Li, Q. Li, M. Zhang, and L. Li, "Hybrid improved capuchin search algorithm for plant image thresholding," *Frontiers in plant science*, vol. 14, p. 1122788, 2023.

[15] S. Jeet, A. Barua, D. K. Bagal, S. Pradhan, S. N. Panda, and S. S. Mahapatra, "Parametric investigation of injection moulding for ldpe using capuchin search algorithm and honey badger algorithm," in *Advances in Functional and Smart Materials*, C. Prakash, S. Singh, and G. Krolczyk, Eds. Singapore: Springer Nature Singapore, 2023, pp. 481–497.

[16] M. Elsayed Abd Elaziz, S. Salima, and R. Ibrahim, "Boosting capuchin search with stochastic learning strategy for feature selection," *Neural Computing and Applications*, pp. 1–20, 03 2023.

[17] M. A. Awadallah, I. Abu-Doush, M. A. Al-Betar, and M. S. Braik, "Chapter 19 - metaheuristics for optimizing weights in neural networks," in *Comprehensive Metaheuristics*, S. Mirjalili and A. H. Gandomi, Eds. Academic Press, 2023, pp. 359–377.

[18] M. Kaveh and S. Mesgari, "Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review," *Neural Processing Letters*, pp. 1–104, 10 2022.

[19] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 97–116, 2017.

[20] E. Kaya, "A comprehensive comparison of the performance of meta-heuristic algorithms in neural network training for nonlinear system identification," *Mathematics*, vol. 10, no. 9, 2022.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[22] H. Faris, I. Aljarah, and S. Mirjalili, "Training feedforward neural networks using multi-verse optimizer for binary classification problems," *Applied Intelligence*, vol. 45, p. 322–332, 09 2016.

[23] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.

[24] T. Y. Lim, "Structured population genetic algorithms: A literature survey," *Artificial Intelligence Review*, vol. 41, p. 385–399, 03 2014.

[25] M. Al-Betar, M. Awadallah, I. Doush, A. Hammouri, M. Mafarja, and Z. Alyasseri, "Island flower pollination algorithm for global optimization," *The Journal of Supercomputing*, vol. 75, p. 5280–5323, 08 2019.

[26] T. Thaher and B. Sartawi, "An experimental design approach to analyse the performance of island-based parallel artificial bee colony algorithm," in *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)*, 2020, pp. 1–7.

[27] D. Izzo, M. Ruciński, and F. Biscani, *The Generalized Island Model*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–169.

[28] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[29] R. Morrison and K. De Jong, "Measurement of population diversity," vol. 2310, 10 2001, pp. 31–41.