Swarm Optimization: A Multi-Agent Approach for Neural Network Optimization with Bacterial Foraging Optimization (BFO) as a Use Case

MOHAMED MOUHAJIR
ENSIAS, Mohammed V University
Rabat, Morocco,
mohamed_mouhajir@um5.ac.ma

MOHAMMED NECHBA
ENSIAS, Mohammed V University
Rabat, Morocco,
mohammed_nechba@um5.ac.ma

Prof. Yasser El Madani El Alami
ENSIAS, Mohammed V University
Rabat, Morocco,
yasser.alami@ensias.um5.ac.ma

***Abstract*—This project investigates the application of a multi-agent system (MAS) combined with neural networks and Bacterial Foraging Optimization (BFO) for optimizing complex problems. The MAS, implemented using Mesa, integrates BFO with neural networks trained using backpropagation ([2], [1]). A comparative analysis is conducted between the MAS approach and traditional backpropagation. The findings shed light on the effectiveness of the MAS framework in improving neural network optimization outcomes.**

***Index Terms*—Multi-agent system (MAS)Bacterial Foraging Optimization (BFO), Neural networks.**

I. Introduction

This project delves into the application of a multi-agent system (MAS) that combines neural networks and Bacterial Foraging Optimization (BFO) to optimize complex problems. In recent years, the development of intelligent systems capable of efficiently solving intricate optimization tasks has garnered increasing interest. While neural networks have proven to be powerful tools for addressing such problems, optimizing their performance remains a challenge.

The optimization landscape for neural networks is often complex and non-linear, posing difficulties for traditional optimization methods like backpropagation in finding the optimal set of network parameters. To overcome this limitation, researchers have turned to nature-inspired algorithms such as Bacterial Foraging Optimization (BFO), which emulate the foraging behavior of bacteria to search for optimal solutions.

In this project, we propose a novel approach that integrates BFO with neural networks within a multi-agent system framework implemented using Mesa. The MAS framework facilitates the coordination and interaction of multiple agents, each representing a unique neural network, in their collective pursuit of the best set of parameters.

To assess the effectiveness of our approach, we conduct a comparative analysis between the MAS framework and traditional backpropagation. We evaluate their performance on complex problems and measure convergence speed and optimization accuracy using various evaluation metrics. Additionally, we explore the influence of factors such as swarm size, bacterial movement, and chemotaxis rates on the optimization process.

By shedding light on the strengths and limitations of the MAS framework combined with BFO, this project aims to contribute to the field of neural network optimization. The findings obtained through this research provide valuable insights into the effectiveness of the MAS approach and its potential for enhancing neural network optimization outcomes.

II. Exploring the BFO Algorithm

A. Overview about BFO

The Bacterial Foraging Optimization (BFO) algorithm is a bio-inspired optimization technique that mimics the foraging behavior of bacteria to solve optimization problems. It is based on the movement and interaction of individual bacteria in a chemotactic process.

The algorithm begins by representing a potential solution as a population of virtual bacteria. Each bacterium corresponds to a potential solution and is characterized by its position and fitness value. The goal is to find the fittest bacterium that represents the optimal solution.

The BFO algorithm consists of several key steps. First, the chemotaxis step simulates the movement of bacteria toward higher nutrient concentrations, which corresponds to better solutions in the optimization problem. During chemotaxis, bacteria move in the solution space by following a biased random walk, with the step size determined by their swimming speed and rotational diffusion. The bacteria evaluate their fitness at each step and update their position accordingly.

After a certain number of chemotaxis steps, the elimination-dispersal step is performed. It simulates the elimination and dispersal of bacteria that have not found promising solutions. Bacteria with lower fitness values are eliminated, and new bacteria are introduced to maintain the population size.

The reproduction step involves the replication of bacteria with higher fitness values. This process creates offspring bacteria that inherit the characteristics of their parent bacteria, with some small modifications introduced through mutation.

Lastly, the hypermutation step introduces random perturbations to the positions of bacteria to promote exploration of the solution space. This step helps prevent the algorithm from converging prematurely to suboptimal solutions.

The BFO algorithm iteratively performs these steps for a predetermined number of generations, aiming to improve the fitness of the population and converge to an optimal solution. By simulating the foraging behavior of bacteria, the algorithm explores the solution space efficiently and has shown effectiveness in solving various optimization problems.

B. The algorithm of BFO

Algorithm 1: Bacterial Foraging Optimization (BFO)

INPUT

- G: Total number of iterations or generations
- N: Number of bacteria
- S: Number of chemotactic steps
- C: Number of elimination-dispersal steps
- E: Number of reproduction steps
- H: Number of hypermutation steps
- $\mathbf{X}$: Initial position of bacteria
- $f(\mathbf{X})$: Objective function

OUTPUT

- $\mathbf{X}^*$: Optimal solution
- $f(\mathbf{X}^*)$: Objective function value at $\mathbf{X}^*$

Initialization:

- Randomly initialize the positions of N bacteria in the search space: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$

Iteration: for $g = 1$ *to* G **do**
  for *each bacterium* $\mathbf{x}_i$ **do**
    Chemotaxis: for $s = 1$ *to* S **do**
      Randomly update the position of bacterium $\mathbf{x}_i$ using chemotactic steps
    end
    Evaluate the objective function $f(\mathbf{x}_i)$
  end
  Elimination-Dispersal: for $c = 1$ *to* C **do**
    Identify bacteria with low fitness and eliminate them.
    Randomly disperse new bacteria in the search space.
  end
  Reproduction: for $e = 1$ *to* E **do**
    Select pairs of bacteria for reproduction
    Perform crossover and mutation
    Evaluate the objective function
  end
  Hypermutation: for $h = 1$ *to* H **do**
    Randomly mutate the positions of bacteria
  end
  Update the positions of bacteria based on fitness values
end
Return $\mathbf{X}^*$ and $f(\mathbf{X}^*)$

C. Flowchart

To gain a better understanding of the operational principles of Bacterial Foraging Optimization (BFO), we will provide the flowchart, as explained in the publication referenced as [1].

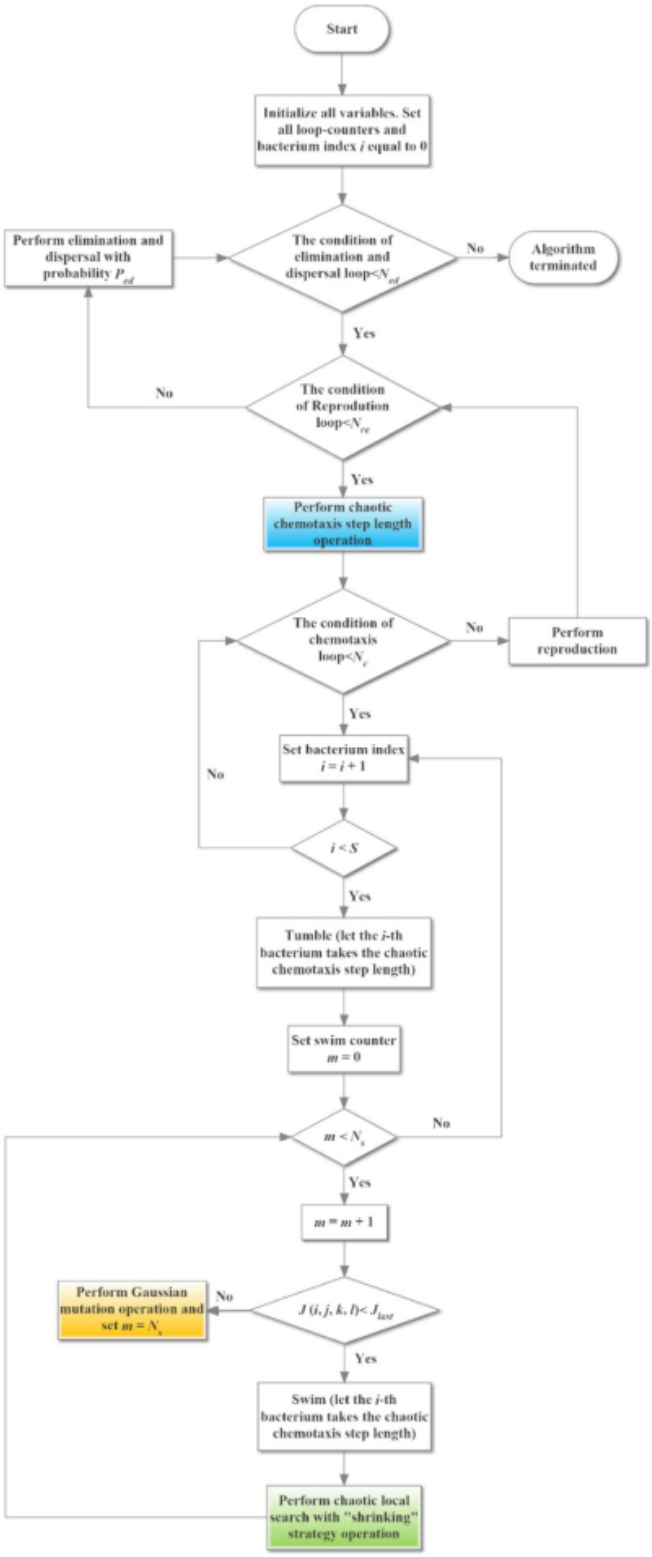

Fig. 1. BFO flowchart

III. Experimentation

A. Experimental Setup

For our experimentation, we generated a random dataset, which comprises a total of 10,000 examples and 70 features. To ensure reliable evaluation.

B. Evaluation Metrics

To assess the performance of the neural network models, we employed the mean squared error (MSE). MSE, quantifies the average squared difference between the predicted and actual values, serving as an indicator of regression performance. By considering both classification and regression aspects, these metrics comprehensively evaluate the effectiveness of the models.

C. Baseline Model: Classical Backpropagation

As a benchmark, we implemented a classical backpropagation model. The model architecture consisted of three fully connected layers with 10, 10, and 1 neurons, respectively. We used the ReLU activation function for the hidden layers and a linear activation function for the output layer. During training, we employed a learning rate of 0.001 and trained the model for 10 epochs. To prevent overfitting, we applied early stopping based on the validation loss, with a patience of 10 epochs.

D. BFO-Based Backpropagation Model

To integrate the BFO algorithm into the backpropagation process, we introduced modifications to the classical backpropagation model. In addition to the standard backpropagation steps, we incorporated the BFO chemotaxis mechanism to optimize the weights of the neural network. We set the number of agents to 10 and performed chemotaxis steps for each agent. The chemotaxis process involved evaluating the objective function (MSE) at different positions in the search space and adjusting the agent's position accordingly. The BFO algorithm introduced an additional hyperparameter, the elimination-dispersion factor, which we set to 0.23.

E. Results and Analysis

we will compare the performance of classical Backpropagation based neural network, and the BFO based Neural Network:

TABLE I
Performance comparison

	MSE	time(seconds)
Gradient Descent based Neural Network	170.25	0.06
BFO based Neural Network	1.65	1.54

The provided table I compares the performance of two neural network training approaches: Gradient Descent based Neural Network and BFO based Neural Network.

The results show that the BFO based Neural Network achieves a significantly lower Mean Squared Error (MSE) value of 1.65 compared to 170.25 for the Gradient Descent based Neural Network. This suggests that the BFO approach provides better accuracy in predicting the target values.

However, it's important to consider the trade-off in terms of training time. The Gradient Descent based Neural Network completes the training process much faster, taking only 0.06 seconds, while the BFO based Neural Network requires 1.54 seconds.

In summary, the BFO based Neural Network outperforms the Gradient Descent based Neural Network in terms of accuracy (lower MSE), but at the cost of increased training time. The choice between the two approaches depends on the specific requirements of the application, balancing the need for accuracy and computational efficiency.

IV. Conclusion

In conclusion, this project explored and compared two approaches for neural network training: the Multi-Agent Bacterial Foraging Optimization (BFO) algorithm and the classical backpropagation technique. The BFO algorithm demonstrated its potential as a nature-inspired optimization technique, while backpropagation showcased its effectiveness in updating neural network weights based on gradients. Further research can be conducted to evaluate the performance of these approaches in various datasets and domains.

References

[1] Huiling Chen, Qian Zhang, Jie Luo, Yueting Xu, and Xiaoqin Zhang. An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine. *Applied Soft Computing*, 86:105884, 2020.

[2] Siddharth Singh Chouhan, Ajay Kaul, Uday Pratap Singh, and Sanjeev Jain. Bacterial foraging optimization based radial basis function neural network (brbfnn) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology. *Ieee Access*, 6:8852–8863, 2018.