Pheasant Foraging Algorithm (PFA): A Bio-Inspired Approach for High-Dimensional Optimization

Jincheng Zhang [1,*], Jindong Zhang [2]
[1] Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand
[2] Independent Researcher, Putian, Fujian 351144, China
Email: [1] zjc1639834588@gmail.com, [2] 172825661@qq.com
*Corresponding Author

***Abstract*—This paper proposes an optimization algorithm based on pheasant foraging behavior, the Pheasant Foraging Algorithm (PFA). The algorithm simulates the collective cooperation and strategy selection of pheasant groups in the foraging process and is used to solve high-dimensional optimization problems. Based on the analysis of pheasant foraging patterns, an adaptive improvement strategy is proposed to improve local search efficiency while maintaining global search capabilities. Experimental results show that compared with classical optimization methods such as particle swarm optimization (PSO) and genetic algorithm (GA), the PFA algorithm has better performance on many standard optimization problems, stronger global search capabilities and more stable convergence performance. The core innovation of PFA lies in its adaptive improvement strategy, which dynamically adjusts search behavior based on environmental feedback to balance global exploration and local exploitation. Unlike PSO and GA, which often suffer from premature convergence or limited local refinement, PFA introduces role-based cooperation and adaptive flight mechanisms inspired by pheasant group foraging behavior.**

Keywords*—*Metaheuristic Algorithm, Algorithm, Optimization Method, Optimization Algorithm

I. INTRODUCTION

Pheasants such as the ring-necked pheasant exhibit complex and efficient behavior patterns during foraging, which are not only affected by the distribution of food resources and environmental factors, but are also closely related to group cooperation. In the natural environment, pheasants make the foraging process more efficient through group cooperation and division of labor. Group members coordinate the division of labor and give full play to their respective advantages to ensure the success of foraging activities. This behavior is not only to meet the needs of survival, but also reflects how species in the ecosystem continuously optimize their own behavior patterns in a resource-constrained environment [1]-[3].

In this context, transforming pheasant foraging behavior into algorithmic ideas not only has strong theoretical significance, but also can provide effective solutions to practical problems. In recent years, heuristic optimization algorithms have been widely used in engineering and scientific research [4]-[10], especially in solving complex problems and optimization tasks [11]-[16]. By simulating the behavior patterns of natural organisms, many optimization algorithms show stronger global search capabilities and adaptability than traditional methods. As a collective behavior pattern in nature, the foraging behavior of pheasants provides valuable inspiration for optimization algorithms. Its collective cooperation and adaptive strategy are the key to solving the problem that traditional optimization algorithms are prone to fall into local optimal solutions.

Optimization algorithms are often used to solve complex engineering problems, machine learning tasks, and various decision-making problems [16]-[22]. However, traditional optimization methods often face the dilemma of local optimal solutions and find it difficult to find the global optimal solution [23]-[28]. To overcome this problem, researchers have proposed many new algorithm ideas, among which a class of algorithms based on biological group behavior has attracted widespread attention. The Pheasant Foraging Algorithm (PFA) is inspired by the collective behavior and adaptive strategy of pheasants in the foraging process, and aims to achieve the search for the global optimal solution by simulating these natural behaviors. While exploring the global optimal solution, the algorithm strives to find the local optimal solution, thereby improving the convergence speed and stability of the algorithm.

Compared with existing bio-inspired algorithms such as PSO, GA, and ant colony optimization (ACO), the PFA emphasizes multi-role group dynamics and dynamic adaptability, rather than relying solely on position updates or probabilistic selection. The choice of pheasant behavior as a model is motivated by their highly organized division of labor, flexible vigilance strategies, and context-aware foraging decisions, which offer richer behavioral diversity than many commonly used biological inspirations. In PFA, group vigilance is abstracted as global information sharing, while individual foraging is mapped to local search with adaptive escape mechanisms, making it suitable for solving complex, multimodal, and high-dimensional optimization problems. Typical application scenarios include engineering design optimization, hyperparameter tuning in machine learning, and resource allocation in uncertain environments.

In summary, the contributions of this work are as follows: A novel bio-inspired algorithm based on pheasant foraging behavior is proposed; An adaptive improvement mechanism is introduced to enhance local exploration; Superior performance is demonstrated over PSO and GA in terms of convergence speed and solution quality;

The algorithm shows strong potential in machine learning and engineering optimization tasks.

II. Foraging Behavior of Pheasants

The foraging behavior of pheasants is part of their survival strategy, which is usually reflected in group foraging, search patterns, foraging strategies, and territory selection. By observing these behaviors, we can find out how they achieve efficient foraging through cooperation and adaptation to the changing environment [29]-[31].

Group foraging: Pheasants usually act in groups and improve foraging efficiency through cooperation and division of labor. During foraging, group members reduce unnecessary waste through clear division of labor. Some individuals are responsible for keeping alert and ensuring that group members can detect threats from predators in time, while others focus on foraging. This division of labor greatly improves foraging efficiency and reduces the risk of excessive competition. In addition, vigilance is also highly adaptable, and they can flexibly adjust their vigilance strategies according to environmental changes and predator activity patterns. Searching method: Pheasants' food sources mainly include plant seeds, fruits, and insects. They use a variety of different foraging techniques to find food, such as pecking at plants on the ground and digging in the soil to find insects or food residues. During foraging, pheasants show strong adaptability and are able to adjust their foraging strategies according to different environmental changes. For example, in seasons when food is scarce, they may choose more insects or other high-energy food sources. In seasons when food is relatively abundant, they may prefer a stable food source, such as plant seeds. Foraging strategy:

Diverse diet: Pheasants adjust their foraging strategies according to food availability and seasonal changes. In summer, they rely more on insects and in winter, they rely more on plant seeds. This strategy helps pheasants cope with seasonal changes and ensures that they get enough nutrition in different ecological environments. Dispersed foraging: When food is unevenly distributed, pheasants will choose to disperse foraging to avoid excessive resource competition. This dispersed behavior not only ensures that each pheasant gets enough food, but also reduces conflicts within the group and improves overall foraging efficiency.

Territory selection: Pheasants usually choose relatively open areas when foraging. Such areas not only provide sufficient food resources, but also effectively avoid the threat of predators. Pheasants optimize resource acquisition by choosing suitable foraging locations while reducing the risk of being preyed on.

These foraging strategies show how individuals in nature optimize resource acquisition through cooperation and adaptation. The collective behavior and adaptation strategies of pheasants in the foraging process provide inspiration for solving optimization problems, especially algorithm design. By simulating the foraging behavior of pheasants, more effective optimization algorithms can be designed to deal with complex problems in reality.

III. Pheasant Foraging Algorithm (PFA)

The Pheasant Foraging Algorithm (PFA) is an optimization algorithm designed based on simulating the foraging behavior of pheasants. During the foraging process, pheasants achieve efficient resource acquisition through mutual cooperation, division of labor and adaptation strategies. These behavioral patterns provide inspiration for the design of new heuristic optimization algorithms. The PFA algorithm is essentially built on these behavioral patterns. The core idea of the PFA algorithm is to find the optimal solution by performing a global search in a vast search space through the interaction between individuals. Specifically, the PFA algorithm mainly adopts the following strategies:

Individual search: In the PFA algorithm, each individual randomly selects an initial position in the search space and then performs a local search according to a random strategy. During the search process, the individual will decide whether to change its position based on the fitness information of its current position. If a more ideal solution is found in the current area, the individual will increase the exploration of the area in an attempt to further optimize the quality of the solution. This process imitates the foraging strategy of pheasants to adjust their foraging strategy according to food abundance. Collective cooperation: Individuals in a group are not completely independent. They will adjust according to the information of the global optimal solution, thereby accelerating the convergence of the global optimal solution. During the foraging process, individuals adjust their behavior through information transmission and mutual cooperation to quickly find the global optimal solution. Similar to how pheasants reduce the threat of predators through vigilance and cooperation, the cooperation between individuals in the PFA algorithm promotes the search process for the global optimal solution.

Adaptive strategy: In the PFA algorithm, when an individual finds that it cannot find a high-quality solution in the current area, it changes its strategy according to certain rules. This strategy can be that the individual flies or moves to a new area, imitating the pheasant's change of foraging location or method during foraging to adapt to environmental changes. Through this adaptive strategy, individuals can avoid stagnation near the local optimal solution, thereby further expanding the search space and increasing the chance of finding the global optimal solution.

By imitating these complex behaviors in nature, the PFA algorithm not only has a strong local search capability, but also effectively avoids falling into the dilemma of the local optimal solution. Therefore, it performs well in solving complex optimization problems.

IV. Algorithm Implementation

The implementation process of the PFA algorithm is as follows:

Initialize the population and calculate the fitness: First, initialize the individuals in the population, randomly generate a position for each individual, and calculate its fitness. The fitness function is usually designed according to the goal of the problem and is used to evaluate the quality of the current solution. Each individual performs a local search based on the current optimal position: In each iteration, the individual performs a local search based on the current optimal solution. The individual will not only adjust the position according to its own fitness, but also refer to the information of the global optimal solution to improve its own solution.

Update the global optimal solution: After each iteration, the algorithm will update the global optimal solution based on the search results of the current population. If a solution is better than the current global optimal solution, the global optimal solution is updated.

In each iteration, the individual will move in the search space according to a random strategy: Each individual will simulate the behavior of pheasants when foraging, search in the search space according to a random strategy, and try to find a better solution. This process not only simulates the autonomous search behavior of individuals, but also reflects the cooperation and information exchange between groups.

Continue to iterate until the maximum number of iterations is reached: The above process will continue to iterate until the preset maximum number of iterations is reached. The PFA algorithm gradually finds the optimal or approximately optimal solution to the problem through multiple iterations.

Through the above steps, the PFA algorithm can effectively search the entire solution space and accelerate the convergence process of the global optimal solution through cooperation between individuals. The pseudocode of the Pheasant Foraging Algorithm (PFA) is as follows:

```
Algorithm: Pheasant Foraging Algorithm (PFA)
Input:
  - pop_size: number of individuals in the population
  - dim: dimensionality of the search space
  - max_iter: maximum number of iterations
  - lb, ub: lower and upper bounds of the search space
Output:
  - gbest_pos: best position found
  - gbest_fit: fitness of the best position
Begin:
  1. Initialize the population X randomly within bounds [lb, ub]
  2. Evaluate fitness f(X) for all individuals
  3. Set gbest_pos = best individual in X
  4. Set gbest_fit = fitness(gbest_pos)
  For iter = 1 to max_iter do:
    For i = 1 to pop_size do:
      - Generate a random number r ∈ [0, 1]
      If r < 0.5 then:
        # Local search (individual exploration)
        X[i] = X[i] + α * randn(dim)   # α: step size
      Else:
        # Social learning (cooperative guidance)
        X[i] = gbest_pos + β * randn(dim)   # β: learning intensity
            # Boundary check
      Clip X[i] within [lb, ub]
    Evaluate f(X) for all individuals
    Update gbest_pos and gbest_fit if better solution is found
    If iter % T_adapt == 0 then:
      # Adaptive behavior trigger
      For i = 1 to pop_size do:
        If fitness of X[i] has not improved in T_adapt:
          - Perform strategy switch:
            X[i] = lb + rand(dim) * (ub - lb)  # Global relocation
  End For
Return gbest_pos, gbest_fit
                End
```

V. Experiments and Results

To evaluate the performance of the PFA algorithm, we compared it with traditional particle swarm optimization (PSO) and genetic algorithm (GA). In order to make a fair comparison, we unified the experimental settings of all algorithms, and the specific experimental settings are as follows:

Group size	: 50 people
Scale	: 10
Maximum number of iterations	: 500
Lower limit	: -10
Upper limit	: 10

In the experiment, we chose the Rosenbrock function as the test function, which is a classic optimization problem and is often used to evaluate the performance of optimization algorithms. The experimental results show that the PFA algorithm has a strong global search capability for multiple standard optimization problems. For example, in the optimization of the Rosenbrock function, the PFA algorithm can quickly converge to the global optimal solution and has stronger stability than PSO and GA.

The researchers conducted the experiment using the following Python code:

```
import numpy as np
# Example objective function
def objective_function(x):
    return np.sum(x**2)  # Example: Rosenbrock function
# PFA (Improved) Algorithm
def PFA_algorithm(pop_size, dim, max_iter, lb, ub):
    # Initialize population
    population = np.random.uniform(lb, ub, (pop_size, dim))
    fitness = np.apply_along_axis(objective_function, 1, population)
    best_position = population[np.argmin(fitness)]
    best_value = np.min(fitness)
        # Improved search strategy
    for iter in range(max_iter):
        for i in range(pop_size):
            r = np.random.rand()
            if r < 0.5:
                population[i] = np.random.uniform(lb, ub, dim)
            else:
                population[i] = best_position + np.random.uniform(-1, 1,
dim)
                # Calculate fitness of the new population
        fitness = np.apply_along_axis(objective_function, 1,
population)
        current_best_value = np.min(fitness)
        current_best_position = population[np.argmin(fitness)]
            # Update global best solution
        if current_best_value < best_value:
            best_value = current_best_value
            best_position = current_best_position
                # Print iteration information
        if iter % 100 == 0:
            print(f"Iteration {iter}: Best value = {best_value}")
    return best_value, best_position
# PSO (Particle Swarm Optimization) Algorithm
def pso_algorithm(pop_size, dim, max_iter, lb, ub):
    # Initialize particle positions and velocities
    population = np.random.uniform(lb, ub, (pop_size, dim))
    velocity = np.random.uniform(-1, 1, (pop_size, dim))
    fitness = np.apply_along_axis(objective_function, 1, population)
        # Initialize personal bests and global best
    personal_best_position = population.copy()
    personal_best_value = fitness.copy()
    global_best_position = population[np.argmin(fitness)]
    global_best_value = np.min(fitness)
    # PSO parameters
    w = 0.5  # Inertia weight
    c1 = 1.5  # Personal learning factor
    c2 = 1.5  # Social learning factor
    for iter in range(max_iter):
        # Update particle velocities and positions
        r1, r2 = np.random.rand(pop_size, dim),
np.random.rand(pop_size, dim)
        velocity = w * velocity + c1 * r1 * (personal_best_position -
population) + c2 * r2 * (global_best_position - population)
        population = population + velocity
```

```
            # Apply position constraints
            population = np.clip(population, lb, ub)
            # Calculate fitness of the new population
            fitness = np.apply_along_axis(objective_function, 1, population)
                # Update personal bests
            for i in range(pop_size):
                if fitness[i] < personal_best_value[i]:
                    personal_best_value[i] = fitness[i]
                    personal_best_position[i] = population[i]
            # Update global best
            current_global_best_value = np.min(fitness)
            if current_global_best_value < global_best_value:
                global_best_value = current_global_best_value
                global_best_position = population[np.argmin(fitness)]
                    # Print iteration information
            if iter % 100 == 0:
                print(f"Iteration {iter}: Best value = {global_best_value}")
        return global_best_value, global_best_position
    # GA (Genetic Algorithm) Algorithm
    def ga_algorithm(pop_size, dim, max_iter, lb, ub):
        # Initialize population
        population = np.random.uniform(lb, ub, (pop_size, dim))
        fitness = np.apply_along_axis(objective_function, 1, population)
        best_position = population[np.argmin(fitness)]
        best_value = np.min(fitness)
            # Improved crossover and mutation strategy
        crossover_rate = 0.8
        mutation_rate = 0.2
        for iter in range(max_iter):
            # Selection operation
            selected_population = population[np.argsort(fitness)[:pop_size // 2]]  # Elite selection
                # If the selected individuals count is odd, duplicate the last individual to make it even
            if len(selected_population) % 2 != 0:
                selected_population = np.append(selected_population, [selected_population[-1]], axis=0)
            # Crossover operation
            offspring = []
            for i in range(0, len(selected_population), 2):
                if np.random.rand() < crossover_rate:
                    crossover_point = np.random.randint(1, dim)
                    offspring1 = np.concatenate([selected_population[i][:crossover_point], selected_population[i+1][crossover_point:]])
                    offspring2 = np.concatenate([selected_population[i+1][:crossover_point], selected_population[i][crossover_point:]])
                    offspring.append(offspring1)
                    offspring.append(offspring2)
                else:
                    offspring.append(selected_population[i])
                    offspring.append(selected_population[i+1])
            # Mutation operation
            for i in range(len(offspring)):
                if np.random.rand() < mutation_rate:
                    mutation_point = np.random.randint(dim)
                    offspring[i][mutation_point] = np.random.uniform(lb, ub)
            # Update population
            population[:len(offspring)] = offspring
            fitness = np.apply_along_axis(objective_function, 1, population)
            current_best_value = np.min(fitness)
            current_best_position = population[np.argmin(fitness)]
                # Update global best
            if current_best_value < best_value:
                best_value = current_best_value
                best_position = current_best_position

            # Print iteration information
            if iter % 100 == 0:
                print(f"Iteration {iter}: Best value = {best_value}")
        return best_value, best_position
    # Run Experiment
    def run_experiment():
        pop_size = 50  # Population size
        dim = 10  # Dimension
        max_iter = 500  # Maximum iterations
        lb = -10  # Lower bound
        ub = 10  # Upper bound
            # Run PFA algorithm
        PFA_best_value, PFA_best_position = PFA_algorithm(pop_size, dim, max_iter, lb, ub)
        print(f"PFA Best Value: {PFA_best_value}, Best Position: {PFA_best_position}")
            # Run PSO algorithm
        pso_best_value, pso_best_position = pso_algorithm(pop_size, dim, max_iter, lb, ub)
        print(f"PSO Best Value: {pso_best_value}, Best Position: {pso_best_position}")
            # Run GA algorithm
        ga_best_value, ga_best_position = ga_algorithm(pop_size, dim, max_iter, lb, ub)
        print(f"GA Best Value: {ga_best_value}, Best Position: {ga_best_position}")
    # Execute Experiment
    run_experiment()
    The output of the code is as follows:
    Iteration 0: Best value = 97.48461343193509
    Iteration 100: Best value = 1.4134294992538783
    Iteration 200: Best value = 1.4134294992538783
    Iteration 300: Best value = 1.4134294992538783
    Iteration 400: Best value = 1.4134294992538783
    PFA Best Value: 1.4134294992538783, Best Position: [-4.90473352 -9.06407938  5.52754148 -9.1971595   3.00848142  0.76088004
     10.02070046 -5.67338656  2.77535331  0.08480374]
    Iteration 0: Best value = 44.11032341270157
    Iteration 100: Best value = 5.9151136467190995e-12
    Iteration 200: Best value = 3.865118493063533e-25
    Iteration 300: Best value = 8.280970613205961e-38
    Iteration 400: Best value = 2.153659878674099e-50
    PSO Best Value: 3.981380262983534e-63, Best Position: [-4.37268442e-33 -1.24832356e-32 -1.74916444e-32 -1.71656502e-32
     1.82219558e-32 -1.72184625e-32 -6.66929063e-33 -3.03956829e-32
     3.58623003e-32  1.79670494e-32]
    Iteration 0: Best value = 105.36690005672688
    Iteration 100: Best value = 11.080224573553508
    Iteration 200: Best value = 11.080224573553508
    Iteration 300: Best value = 11.080224573553508
    Iteration 400: Best value = 11.080224573553508
    GA Best Value: 11.080224573553508, Best Position: [-3.83923988 -0.37177041 -9.35825837  1.25712549  0.08802301 -2.61725471
     0.21017754  0.8196534  -1.34227676 -1.5193126 ]
```

The output of this code shows the performance of three optimization algorithms (PFA, PSO, and GA) on the same problem. Let's analyze the results of each algorithm one by one, focusing on the advantages of the PFA algorithm.

1. PFA (Pheasant Foraging Algorithm) Result Analysis: In the first iteration, the optimal value is 97.48.

After 100 iterations, the optimal value drops significantly to 1.41 and remains until 400 iterations. The final optimal value is 1.41, and the corresponding position is [-4.90473352,-9.06407938,5.52754148,-9.1971595,...].

- Advantages of the PFA algorithm:

Fast convergence speed: From 97.48 in the first iteration to 1.41 in the 100th iteration, PFA quickly converges to the optimal solution in the early stage and stabilizes near the optimal value, showing a strong global search ability.

Good stability: The optimal value basically does not fluctuate during the iteration process, indicating that the algorithm is more stable during the search process.

Able to find an effective solution: The final optimal value is 1.41, indicating that PFA can effectively find a solution close to the global optimal.

2. Particle Swarm Optimization (PSO) Result Analysis: In the first iteration, the optimal value is 44.11.

As the number of iterations increases, the optimal value decreases rapidly and eventually approaches the minimum value (for example, 3.98e-63).

The final optimal value is 3.98e-63, and the corresponding value is close to zero.

- Problems with the particle swarm optimization algorithm:

Convergence speed is too fast: The value of the particle swarm optimization algorithm tends to zero in the later stage, indicating that it may fall into a local optimal solution too early.

Accuracy problem: Because the final value is too small, there may be problems with the accuracy of the numerical calculation, making the final solution meaningless.

3. Genetic Algorithm (GA) Result Analysis: In the first iteration, the optimal value is 105.37.

After 100 iterations, the optimal value dropped to 11.08 and remained there until the 400th iteration.

The final optimal value is 11.08, and the corresponding position is [-3.83923988,-0.37177041,-9.35825837,1.25712549,...].

- Problems with the GA algorithm:

Slow convergence: After multiple iterations, the optimal value of the GA algorithm has not been significantly improved, and the convergence speed is slow. It may take more iterations to find a better solution.

Final solution difference: Although the number of iterations reached 400 times, the optimal value of the GA algorithm was only 11.08, which is much higher than the 1.41 of the PFA algorithm, indicating that the quality of its final solution is poor.

- About:

Compared with the particle swarm optimization algorithm and the genetic algorithm, the PFA algorithm shows better performance in terms of fast convergence and stability. The PFA algorithm can find a solution close to the global optimal solution in a short time and stabilize near the optimal value, which makes it perform well in optimization problems.

Although the PSO algorithm can find a smaller solution, it may be meaningless if it converges to near zero too early, and the numerical accuracy is low.

The GA algorithm converges slowly, and the final optimal value is not as good as the PFA algorithm.

Therefore, the advantages of the PFA algorithm are fast convergence speed, good stability, and the ability to find an effective solution close to the global optimal solution.

VI. Conclusion

The pheasant foraging algorithm (PFA) proposed in this paper provides a new method for solving optimization problems by simulating the group behavior and adaptation strategy of pheasants foraging. The experimental results show that the PFA algorithm has a strong global search ability and a fast convergence speed. Compared with the traditional particle swarm optimization algorithm (PSO) and genetic algorithm (GA), the PFA algorithm can more effectively avoid the trap of local optimal solutions and find the global optimal solution. Future research will further refine the local search strategy of the PFA algorithm, explore its application in more complex optimization problems, and expand its applicability in engineering, machine learning, and other fields. We hope to provide a better optimization method for solving more complex practical problems by further improving the efficiency and robustness of the PFA algorithm.

References

[1] M. R. Ashrafzadeh *et al*., "Assessing the origin, genetic structure and demographic history of the common pheasant (Phasianus colchicus) in the introduced European range," *Scientific Reports*, vol. 11, no. 1, p. 21721, 2021, https://doi.org/10.1038/s41598-021-00567-1.

[2] A. Uçar, "The effects of stocking density on growth, morphological development, behavior, and welfare parameters in pheasants (Phasianus colchicus)," *Tropical Animal Health and Production*, vol. 56, no. 1, p. 18, 2024, https://doi.org/10.1007/s11250-023-03856-1.

[3] T. R. Shirley and A. K. Janke, "Ring-necked pheasant nest site selection in a landscape with high adoption of fall-seeded cover crops," *Wildlife Society Bulletin*, vol. 47, no. 1, p. e1394, 2023, https://doi.org/10.1002/wsb.1394.

[4] B. Abdollahzadeh *et al*., "Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning," *Cluster Computing*, vol. 27, no. 4, pp. 5235-5283, 2024, https://doi.org/10.1007/s10586-023-04221-5.

[5] M. A. Elaziz *et al*., "Advanced metaheuristic optimization techniques in applications of deep neural networks: a review," *Neural Computing and Applications*, vol. 33, pp. 14079-14099, 2021, https://doi.org/10.1007/s00521-021-05960-5.

[6] B. Benaissa, M. Kobayashi, M. Al Ali, T. Khatir, and M. E. A. E. Elmeliani, "Metaheuristic optimization algorithms: An overview," *HCMCOU Journal of Science–Advances in Computational Structures*, vol. 14, no. 1, pp. 33-61, 2024, https://doi.org/10.46223/HCMCOUJS.acs.en.14.1.47.2024.

[7] R. Alkanhel, E. S. M. El-kenawy, D. L. Elsheweikh, A. A. Abdelhamid, A. Ibrahim, and D. S. Khafaga, "Metaheuristic optimization of time series models for predicting networks traffic," *CMC-Computers, Materials & Continua*, vol. 75, no. 1, pp. 427-442, 2023, https://doi.org/10.32604/cmc.2023.032885.

[8] M. Kaveh and M. S. Mesgari, "Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review," *Neural Processing Letters*, vol. 55, no. 4, pp. 4519-4622, 2023, https://doi.org/10.1007/s11063-022-11055-6.

[9] O. O. Akinola, A. E. Ezugwu, J. O. Agushaka, R. A. Zitar, and L. Abualigah, "Multiclass feature selection with metaheuristic optimization algorithms: a review," *Neural Computing and Applications*, vol. 34, no. 22, pp. 19751-19790, 2022, https://doi.org/10.1007/s00521-022-07705-4.

[10] P. Sharma, S. Raju, "Metaheuristic optimization algorithms: A comprehensive overview and classification of benchmark test functions," *Soft Computing*, vol. 28, no. 4, pp. 3123-3186, 2024, https://doi.org/10.1007/s00500-023-09276-5.

[11] M. Alyami *et al*., "Application of metaheuristic optimization algorithms in predicting the compressive strength of 3D-printed fiber-reinforced concrete," *Developments in the Built Environment*, vol. 17, p. 100307, 2024, https://doi.org/10.1016/j.dibe.2023.100307.

[12] S. Sankarananth, M. Karthiga, E. Suganya, S. Sountharrajan, and D. P. Bavirisetti, "AI-enabled metaheuristic optimization for predictive management of renewable energy production in smart grids," *Energy Reports*, vol. 10, pp. 1299-1312, 2023, https://doi.org/10.1016/j.egyr.2023.08.005.

[13] O. N. Oyelade, A. E. -S. Ezugwu, T. I. A. Mohamed and L. Abualigah, "Ebola Optimization Search Algorithm: A New Nature-Inspired Metaheuristic Optimization Algorithm," *IEEE Access*, vol. 10, pp. 16150-16177, 2022, https://doi.org/10.1109/ACCESS.2022.3147821.

[14] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Gazelle optimization algorithm: a novel nature-inspired metaheuristic optimizer," *Neural Computing and Applications*, vol. 35, no. 5, pp. 4099-4131, 2023, https://doi.org/10.1007/s00521-022-07854-6.

[15] N. S. Bajaj, A. D. Patange, R. Jegadeeshwaran, S. S. Pardeshi, K. A. Kulkarni, and R. S. Ghatpande, "Application of metaheuristic optimization based support vector machine for milling cutter health

monitoring," *Intelligent Systems with Applications*, vol. 18, p. 200196, 2023, https://doi.org/10.1016/j.iswa.2023.200196.

[16] S. K. Towfek, N. Khodadadi, "Deep convolutional neural network and metaheuristic optimization for disease detection in plant leaves," *Journal of Intelligent Systems and Internet of Things*, vol. 10, no. 1, pp. 66-75, 2023, https://doi.org/10.54216/JISIoT.100105.

[17] S. M. Qaisar, S. I. Khan, K. Srinivasan, and M. Krichen, "Arrhythmia classification using multirate processing metaheuristic optimization and variational mode decomposition," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 1, pp. 26-37, 2023, https://doi.org/10.1016/j.jksuci.2022.05.009.

[18] V. Kumar, and S. M. Yadav, "A state-of-the-art review of heuristic and metaheuristic optimization techniques for the management of water resources," *Water Supply*, vol. 22, no. 4, pp. 3702-3728, 2022, https://doi.org/10.2166/ws.2022.010.

[19] M. Pluhacek, A. Kazikova, T. Kadavy, A. Viktorin, and R. Senkerik, "Leveraging large language models for the generation of novel metaheuristic optimization algorithms," *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 1812-1820, 2023, https://doi.org/10.1145/3583133.3596401.

[20] N. A. Samee *et al*., "Metaheuristic Optimization Through Deep Learning Classification of COVID-19 in Chest X-Ray Images," *Computers, Materials & Continua*, vol. 73, no. 2, pp. 4193-4210, 2022, https://doi.org/10.32604/cmc.2022.031147.

[21] A. H. Halim, I. Ismail, and S. Das, "Performance assessment of the metaheuristic optimization algorithms: an exhaustive review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 2323-2409, 2021, https://doi.org/10.1007/s10462-020-09906-6.

[22] M. M. Eid, F. Alassery, A. Ibrahim, and M. Saber, "Metaheuristic optimization algorithm for signals classification of electroencephalography channels," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 4627-4641, 2022, https://doi.org/10.32604/cmc.2022.024043.

[23] H. B. Ly, M. H. Nguyen, and B. T. Pham, "Metaheuristic optimization of Levenberg–Marquardt-based artificial neural network using particle swarm optimization for prediction of foamed concrete compressive strength," *Neural Computing and Applications*, vol. 33, no. 24, pp. 17331-17351, 2021, https://doi.org/10.1007/s00521-021-06321-y.

[24] S. Ali, S. Riaz, X. Liu, and G. Wang, "A Levenberg–Marquardt based neural network for short-term load forecasting," *Computers, Materials and Continua*, vol. 75, no. 1, pp. 1783-1800, 2023, https://doi.org/10.32604/cmc.2023.035736.

[25] R. K. Pattnaik, M. Siddique, S. Mishra, D. J. Gelmecha, R. S. Singh, and S. Satapathy, "Breast cancer detection and classification using metaheuristic optimized ensemble extreme learning machine," *International Journal of Information Technology*, vol. 15, no. 8, pp. 4551-4563, 2023, https://doi.org/10.1007/s41870-023-01533-y.

[26] S. K. Parhi, S. K. Panigrahi, "Alkali–silica reaction expansion prediction in concrete using hybrid metaheuristic optimized machine learning algorithms," *Asian Journal of Civil Engineering*, vol. 25, no. 1, pp. 1091-1113, 2024, https://doi.org/10.1007/s42107-023-00799-8.

[27] L. Abualigah *et al*., "Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results," *Neural Computing and Applications*, vol. 34, pp. 4081-4110, 2022, https://doi.org/10.1007/s00521-021-06747-4.

[28] E. Pira, "City councils evolution: a socio-inspired metaheuristic optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 9, pp. 12207-12256, 2023, https://doi.org/10.1007/s12652-022-03765-5.

[29] A. Estienne *et al*., "The influence of selection in wild pheasant (Phasianus colchicus) breeding on reproduction and the involvement of the chemerin system," *Poultry Science*, vol. 102, no. 1, p. 102248, 2023, https://doi.org/10.1016/j.psj.2022.102248.

[30] W. Li *et al*., "Widespread arboreal foraging behavior in ground-dwelling birds and the urgency of life-history studies," *Biological Conservation*, vol. 286, p. 110320, 2023, https://doi.org/10.1016/j.biocon.2023.110320.

[31] H. Li, X. Liu, Z. Han, S. Wu, and C. Cao, "Foraging and day-roosting sites selection by the endangered Brown-eared Pheasant Crossoptilon mantchuricum during autumn in the Huanglong Mountains, Shaanxi Province, China," *Pakistan Journal of Zoology*, vol. 53, pp. 401-800, 2021, https://dx.doi.org/10.17582/journal.pjz/20191130101121.