ORIGINAL RESEARCH ARTICLE | UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Enhancing Pelican Optimization Algorithm with Differential Evolution: A Novel Hybrid Metaheuristic Approach

Rebin Abdulkareem Hamaamin[1], Omar Mohammed Amin Ali[2]

[1]Department of Computer Science, College of Sciences, Charmo University, Chamchamal, Sulaymaniyah, Iraq.
[2]Department of Information Technology, Chamchamal Technical Institute, Sulaimani Polytechnic University, Sulaymaniyah, Iraq.

ABSTRACT

In the field of solutions for composite objective functions, the problem of identifying a proper trade-off between exploitation and exploration is still urgent. Classical methods can hardly avoid early iteration convergence or be insufficient in terms of searching throughout the space of potential solutions, especially when dealing with multi-variate multi-dimensional problems. To overcome this problem, this work proposes a combination of the pelican optimization algorithm (POA) and differential evolution (DE), known as the POA-DE metaheuristic method, which comprises the explorative characteristic of POA and the exploitative feature of DE. The main issue dealt with in this work relates to the conflict of global search and local exploitation in the context of solving complex optimization tasks. In global exploration, the POA technique is applied to improve the performances of the search in the large area, and the DE method is used in the local search space for improving the solution. To this end, the proposed solution hybrid model tries to avoid the shortcomings associated with using either of the two key aspects when used independently. To support the results obtained through POA-DE, it is necessary to perform the intensive empirical examination of several benchmark functions. The results also show that the proposed method has achieved better stability, efficiency, and convergence speed than the basic POA. Therefore, extending the hybrid optimization techniques is significant in enhancing the meta-heuristic algorithms that form a powerful tool to solve the optimization problems.

Index Terms: Pelican Optimization Algorithm, Differential Evolution, Hybrid Metaheuristic, Exploration and Exploitation, Optimization Benchmark Functions

1. INTRODUCTION

Optimization plays a key role in solving complex real-world problems across various disciplines, including engineering, medicine, and economics. One of the most pressing challenges in this domain is finding a suitable balance between exploration (searching new regions of the solution space) and exploitation (refining current solutions). Many metaheuristic algorithms suffer from premature convergence or inefficient space traversal, particularly when handling high-dimensional or multi-modal functions. This paper addresses this critical issue by proposing a hybrid approach that combines the pelican optimization algorithm (POA), known for its strong global search capabilities, with differential evolution (DE), a method renowned for effective local optimization. By combining these two useful methods, the proposed POA-DE algorithm seeks to improve results in tackling difficult optimization problems and to get past the drawbacks of using each method on its own. Optimizing complex non-linear functions with constraints remains a critical challenge in engineering and computational science.

Access this article online

DOI:10.21928/uhdjst.v9n2y2025.pp101-114	**E-ISSN:** 2521-4217 **P-ISSN:** 2521-4209

Corresponding author's e-mail: Rebin Abdulkareem Hamaamin, Department of Computer Science, College of Sciences, Charmo University, Chamchamal, Sulaymaniyah, Iraq. E-mail: rebin.abdulkarim@chu.edu.iq

Received: 30-04-2025 Accepted: 14-08-2025 Published: 08-09-2025

Existing metaheuristics often face a trade-off between exploration and exploitation, which limits their effectiveness in high-dimensional and noisy environments. This paper addresses this challenge by proposing a novel hybrid algorithm that balances these aspects to achieve improved optimization performance.

This paper proposes a novel hybrid metaheuristic, the POA-DE, which synergistically integrates the exploratory capabilities of POA with the exploitative advantages of DE. This integration leads to improved convergence speed and solution quality on complex optimization problems compared to the original POA and other metaheuristics.

This optimization is the systematic process of finding the most favorable solution within a reasonable timeframe. This region has seen significant changes with the introduction of a genetic algorithm (GA) and DE. As a result, the number of optimization challenges is increasing in complexity. Therefore, the resolution of these issues requires the application of more efficient optimization techniques [1]. There are several effective algorithms available for solving a certain issue. Nevertheless, it is premature to designate any one of them as the superior option until this study has conducted a comprehensive evaluation of their relative performance in addressing the problem under consideration. Optimization algorithms are capable of efficiently solving many difficulties [2]. Optimization problem-solving methodologies can broadly be classified into two group's deterministic methods and stochastic methods [3]. When trying to solve difficult optimization problems with objective functions that are discontinuous, high-dimensional, non-convex, and non-derivative, deterministic approaches have trouble. Stochastic methods, unlike deterministic methods, can effectively address the challenges of optimization problems by utilizing random search in the problem-solving space. These methods do not rely on derivative and gradient information from the optimization problem's objective function [4]. People often classify stochastic as a heuristic or metaheuristic. Nature-inspired metaheuristic algorithms are capable of efficiently solving both real-world issues and traditional mathematical functions throughout their exploration and exploitation stages. However, achieving a balance between these two stages is a critical challenge that metaheuristic optimizations struggle with [5]. Various optimization issues have been tackled using metaheuristic methods. The objective of using these algorithms is to determine the maximum or lowest value of a certain function, such as minimizing the time required for a specific journey or minimizing the cost of completing a task [6]. However, these algorithms do have some shortcomings when it comes to achieving global optima since they must balance the competing goals of exploration and exploitation. Due to their excellent performance, metaheuristic algorithms tackle real-world issues. The problem appears to have its roots in electromagnetics [7], engineering design problems [8], constrained optimization problems [9], economic problems [10], medical problems [11], and task planning problems [12]. These methods are effectively utilized in a wide range of engineering and scientific applications, such as optimizing power generation in electrical engineering, designing bridges and buildings in civil engineering, performing data mining tasks such as classification, prediction, clustering, and system modeling, as well as designing radars and networking systems in communication [13].

In 2022, Trojovský and Dehghani [14] introduced a new optimization technique called the Pelican optimization technique POA. They have modeled the design after the foraging actions of pelicans. Compared to eight well-known SI optimization methods, the POA achieves outstandingly comparable performance by effectively balancing exploration and exploitation. Therefore, many practical applications utilize the POA. Although the standard POA is valuable, it is also susceptible to local optimization. To overcome this problem, several academics have proposed alternate, improved methodologies. They implemented tent chaos to improve population diversity and incorporated a dynamic weight factor to enable the pelican's continuous position updates. These approaches surpass classic POAs in performance and provide better outcomes in 10 benchmark functions. However, they did not compare the execution times of various algorithms [15]. Each particle modifies its path toward its past optimal position and the current optimal position achieved by any other member in its local area [16]. Particle swarm optimization (PSO) has the advantage over GA in that it is theoretically straightforward, requires little calculation time, and has a limited number of parameters to change. Nevertheless, the primary drawback of PSO is the potential for premature search convergence, particularly in intricate multi-peak search issues. In this study have developed a hybrid approach that combines PSO with DE to address this issue and enhance the efficiency of the PSO algorithm, this work proposes the PSO-DE method. DE is an enhanced iteration of GA, first introduced by Abualigah *et al.* [17]. A hybrid PSO-DE technique is proposed to tackle a global optimization problem. A hybrid PSO-DE technique is presented, combining the speed of PSO with the exploration capabilities of DE. The hybrid approach employs the PSO algorithm to identify the best solution area, followed by a mix of PSO and DE algorithms to locate the ideal point [18]. The POA-DE distinguishes itself from previous POAs because it uses DE in its hybridization process. This way, the hybridization process can combine the characteristics of DE such as mutation and crossover with those of the Population-based Optimization Algorithm POA, which works by repeating itself. The proposed hybrid method incorporates an additional DE phase into the primary loop of the PSO. This novel addition aims to enhance both the exploration and exploitation capabilities of the original POA. The integration of DE's methodology for generating a pool of prospective solutions and POA's methodology for modifying positions led to the development of POA-DE, which offers a more balanced and comprehensive approach to global search and local fine-tuning. The POA-DE is a novel kind of hybridization that significantly enhances the optimization outcomes of the original POA.

Our research is organized around a POA framework, with its mechanism detailed in Section 2. Section 2 then elucidates the POA Work and Flowchart. Section 3 describes DE. Section 4 provides a comprehensive explanation of our proposed POA-DE approach. Section 5 involves a comparison between POA-DE and 23 benchmark test functions [19], statistical results are documented and presented. Finally, the conclusion is presented, and suggestions for future research are offered.

2. LITERATURE REVIEW

Stochastic population-based optimization algorithms are among the best methods for addressing optimization problems. Based on the primary concepts and sources of inspiration that shaped their design, optimization algorithms may be broadly divided into four groups: Swarm-based optimization techniques that are game-, physics-, and evolutionary-based. Natural phenomena, such as the swarm behaviors of insects, animals, and other living things, are considered while developing swarm-based optimization algorithms. One of the first and most widely used swarm-based algorithms is PSO, which draws inspiration from how birds forage for food. The best position each population member has encountered and the best position the whole population has experienced are used to update each member's status in the PSO [20]. The modeling of a classroom environment and student-teacher interactions serves as the foundation for teaching-learning-based optimization (TLBO). Population members in TLBO exchange information with one another and receive updates as part of teacher training [21]. The social behavior and hierarchical structure of gray wolves while hunting serve as the inspiration for gray wolf optimization (GWO). Alpha, beta, delta, and omega wolves are the four wolf types employed in GWO to simulate the hierarchical leadership of gray wolves. Simulations update population members by modeling the three primary hunting stages: Searching for prey, surrounding prey, and attacking prey [22]. The Whale optimization algorithm (WOA) is a swarm-based optimization algorithm inspired by nature that models the social behavior of humpback whales and their bubble-net hunting technique. WOA uses three hunting phases searching for prey, surrounding prey, and humpback whale bubble-net foraging behavior to update population members [23]. In this research, a Tunicate Swarm Algorithm (TSA) is developed by simulating the swarm behavior and jet propulsion of tunicates during feeding and navigation. Four phases avoidance of search agent (SA) conflicts, convergence toward the best SA, movement toward the best neighbor, and swarm behavior are used by TSA to update the population [24]. The movement strategies used by marine predators to catch their food in the ocean served as the model for the marine predators algorithm (MPA). The population update process in MPA is divided into three stages due to the different speeds of the predator and prey: (i) The predator is faster, (ii) the predator and prey are equal in speed, and (iii) the prey is faster [25]. The introduction of evolutionary-based optimization algorithms is predicated on models of genetic, biological, and other evolutionary processes. One of the first and most popular evolutionary algorithms is the GA, which draws inspiration from Charles Darwin's idea of natural selection and the reproductive process. In [26], the authors employ three primary operators' selection, crossover, and mutation to update the population members. The artificial immune system (AIS) algorithm, a revolutionary approach, is based on the immune system's response to viruses and bacteria. Cognitive, activation, and effector stages all have an impact on the population updating process in AIS [27]. The modeling of the many physics laws serves as the foundation for the development of physics-based optimization methods. The metallurgical melting and cooling process inspired the physics-based approach known as "simulated annealing." To lessen its flaws, the material is heated and then softly cooled under carefully monitored circumstances. The SA optimizer was designed using mathematical modeling of this process [28]. The modeling of the gravitational attraction between objects at varying distances from one another served as the inspiration for the gravitational search algorithm (GSA). The GSA updates its population members by modeling Newtonian principles of motion and calculating gravitational force [29]. The construction of game-based optimization algorithms is based on modeling player behavior and the rules of various solo and multiplayer games. A game-based algorithm called the football game-based optimizer (FGBO) simulates player behavior and club relations in a football league. The four stages of league holding, training, player transfers between teams, and club promotion and relegation form the basis of FGBO's population update procedure [30]. The foundation of Tug of War Optimization (TWO) is modeling player behavior in a tug of war. Modeling the tensile force between population members who compete with one another serves as the foundation for TWO's population member update method [31].

While numerous metaheuristic algorithms such as GA, PSO, and DE have been proposed, most prior studies provide primarily descriptive overviews of their applications. A critical comparison reveals that many of these methods suffer from premature convergence or lack the balance between exploration and exploitation necessary for complex, high-dimensional optimization problems. The proposed POA-DE addresses this gap by integrating the exploratory nature of the POA with DE's exploitative strengths, aiming to improve convergence speed and solution quality.

3. THE POA ALGORITHM AND DE

3.1. Biological Inspiration and Mathematical Modeling of POA

The pelican is a large avian creature known for its prominent beak and throat pouch, which it uses to capture and consume food [32]. This species congregates in communal roosts, where groups of pelicans, sometimes numbering in the hundreds, meet together. We can describe the appearance of pelicans as follows: Pelicans have a weight range of approximately 2.75–15 kg, a height range of 1.06–1.183 m, and a wing span of 0.5–3 m. They primarily consume fish but also feed on frogs, turtles, and other crustaceans. In times of extreme hunger, they may even consume shellfish. Pigeons, particularly pelicans, often exhibit collective behavior when foraging for their prey. On locating their prey, they swiftly submerge into the water, descending a distance of 10–20 m. Indeed, some predators descend to lower regions to capture their prey. Pelicans are considered to be very skilled hunters based on their intellect, hunting habits, and techniques. The suggested approach's modeling primarily inspires the design of the intended POA [14], [33].

The proposed POA is population-based, including pelicans as well. Within population-based algorithms, every individual serves as a prospective solution. Every individual in the population suggests values for the variables of the optimization problem depending on their location in the search space. At first, individuals in the population are allocated randomly within the specified range of values using Equation (1).

$$X_{ab} = Y_b + rand.\left(Z_b - Y_b\right), a = 1,2,\ldots\ldots,M,\ b = 1,2\ldots\ldots N \tag{1}$$

where X_{ab} is the value of the bth variable specified by the ath candidate solution, M is the number of population members, N is the number of problem variables, rand is a random number in interval [0, 1], Y_b is the bth lower bound, and Z_b is the bth upper bound of problem variables. The suggested POA replicates pelican behavior and strategy for approaching and hunting prey, updating potential solutions.

3.2. DE and its Role in POA-DE

The DE algorithm, introduced by Storn and Price [17], is a powerful evolutionary optimization technique widely used for solving continuous and multimodal optimization problems. DE operates through three primary operations: Mutation, crossover, and selection. These steps allow the population to evolve better solutions over iterations.

In the context of the hybrid POA-DE algorithm, DE enhances the global search ability of POA by diversifying the candidate solutions early in the search process, helping avoid premature convergence. The new trial vector is generated using DE's mutation and crossover operations. Depending on the nature of the problem, various mutation and crossover strategies can be employed [34] [35].

This hybridization leverages the global exploration strength of DE and the problem-specific exploitation ability of POA, leading to a more robust and efficient optimization process.

3.3. Parameter Settings

Table 1 summarizes the key parameters used in the POA-DE algorithm. These parameters control the behavior and performance of the algorithm during optimization. Selecting appropriate values is essential to balance exploration of the search space and convergence speed, ensuring effective and efficient optimization results.

4. HYBRID APPROACH: POA-DE

The suggested study introduces a novel metaheuristic algorithm called POA-DE, which combines the features of

TABLE 1: Parameter settings

Parameter	Description	Value	Justification
Population size	Number of pelican agents	30	Balances search diversity and computational cost
Maximum iterations	Max iterations for algorithm	500	Empirically sufficient for convergence in tests
DE mutation factor (F)	Controls mutation step size	0.5	Standard value that balances exploration and exploitation
DE crossover rate	Probability of crossover in DE	0.9	Encourages recombination of solutions

DE: Differential evolution

POA and DE. The purpose of this hybridization is to merge DE's global search capability with POA's local exploitation capability to achieve a balance between exploration and exploitation in the optimization process. The POA-DE algorithm operates in two distinct stages. The first stage employs the DE algorithm. Therefore, DE starts by selecting three pairs of distinct candidate solutions from the population in a random manner. The mutation process generates a third candidate solution by calculating the weighted difference between the first two solutions and adding it to the third solution. This modified candidate subsequently mates with the aforementioned candidate, resulting in the creation of a novel solution. This implies that the population includes the new solution only if its fitness exceeds that of the current population. This phase fortifies the algorithm's learning capabilities as it searches for the optimal solution to a given issue, shielding it from becoming stuck in suboptimal solutions. The POA assumes the role during the second phase. Pelicans' hunting technique is the source of the term "POA." During exploration, they categorize each potential solution as either moving closer to or farther away from a reference solution, also known as a food source. This categorization is based on the fitness value compared to the reference values. Put simply, the candidate approaches the reference point if its fitness is greater, and it goes away if the candidate's ability is lower. It is particularly helpful in investigating the search space. Following the exploration phase, the exploitation phase improves the solutions by including a small random increment that progressively decreases during the exploitation phase. Optimizing this fine-tuning process is crucial for enhancing the exploration of the local search space and improving the quality of the acquired solutions.

The hybrid POA-DE algorithm has been tested using a collection of benchmark functions (F1–F23) to compare the results with the original POA. Experiments were conducted to evaluate the performance and adaptability of POA-DE across various population sizes and iteration counts. The outcomes assessment demonstrated that POA-DE exhibited a superior level of accuracy when compared to the original POA in several functions, thereby confirming the successful integration of DE into the POA framework.

In the DE phase of the proposed POA-DE algorithm, the mutation factor (F) and crossover rate (CR) play crucial roles in controlling the search dynamics. In this study, F is set to 0.5 and CR to 0.9, which are commonly used default values known to provide a good trade-off between exploration and exploitation. To ensure the robustness of these parameters, a sensitivity analysis was conducted by varying F within the range [0.4, 0.9] and CR within [0.5, 1.0]. The results confirmed that the algorithm maintains stable performance under these variations, supporting the suitability of the chosen parameter values for the benchmark functions used in this work.

Algorithm (1) displays the Pseudocode and flowchart for POA-DE.

4.1. Mathematical Equations for Hybrid POA-DE

Equation (2) in the DE phase, which is also called the mutation equation, yields a new candidate solution by introducing diversity into the population. This step is done for three randomly selected solutions, though adding to the third solution a scaled difference between two other solutions, this process creates a mutant vector that has the potential to explore a new area of the search space. Consequently, it aids the algorithm in evading local optima and amplifies its capacity for global search.

$$V = X_a + \mathrm{F}.(\mathrm{X\_b} - \mathrm{X\_c}) \qquad (2)$$

Here, V is the mutant vector, F is the differential weight, X_a+F.(X_b−X_c) are three randomly selected individuals from the population.

Crossover Equation (3) uses the generated mutant vector during the previous step and combines it with the current solution to generate a trial solution. This is achieved through random selection of some of the components either from the mutant vector or the current solution. This process also guarantees that the trial solution has some of the features

ALGORITHM 1: Pseudo code of pelican optimization algorithm-differential evolution

```
Algorithm: POA-DE
1. Input:
  a. Define the objective function f (x)
  b. Set bounds for each decision variable
  c. Set population size N and maximum iterations T
2. Initialization:
  a. Randomly initialize positions of N pelicans within bounds → X
     = {x₁, x₂, ., xₙ}
  b. Evaluate fitness of each pelican: f (xᵢ) ∀ i ∊ [1, N]
3. For t = 1 to T do:
  a. Step 1: Update global best
    i. Identify best pelican: X_best = argmin (f (xᵢ))
    ii. Store best fitness value: f_best = f (X_best)
  b. Step 2: Differential Evolution Phase (DE Phase)
    i. Set DE parameters: F (mutation factor), CR (crossover rate)
    ii. For each pelican i = 1 to N do:
      1. Mutation:
        - Randomly select three distinct pelicans: x_r1, x_r2, x_r3
        - Compute mutant vector: vᵢ = x_r1 + F * (x_r2 - x_r3)
      2. Crossover:
        - Generate trial vector uᵢ by mixing vᵢ and xᵢ based on CR
      3. Selection:
        - If f (uᵢ) < f (xᵢ), then xᵢ ← uᵢ
  c. Step 3: Pelican Optimization Algorithm Phase (POA Phase)
    i. Select a random pelican as prey: X_FOOD
    ii. Phase 1: Exploration (Moving Towards Prey)
      For each pelican i = 1 to N:
        - Update position of xᵢ based on movement toward X_FOOD
    iii. Phase 2: Exploitation (Winging on Water Surface)
      For each pelican i = 1 to N:
        - Refine position of xᵢ using water surface dynamics
  d. Step 4: Update and store best solution found in iteration t
4. Output:
  - Best candidate solution X_best and corresponding fitness f_best
End Algorithm
```

of the original and the mutant solutions, hence diversifying the population to increase the possibility of arriving at an even better solution.

$$U_{i(j)} = \begin{cases} X_{i(j)} & \text{if } rand(j) \le CR \text{ or } j = j_{rand} \\ X_{i(j)} & \text{otherwise} \end{cases} \quad (3)$$

$U_{(i(j))}$ is the trial vector, $X_{(i(j))}$ is the target vector, CR is the crossover rate, and jrand is a randomly chosen index.

The exploration phase in Equation (4) determines the direction of moving a candidate solution in the search space forward or backward depending upon whether the current candidate solution performs better or worse than a randomly selected "food." If the solution becomes worse, it approaches the food in an attempt to better itself. If it's better, it steps slightly back, which makes one search for other options with more fervor. This allows the algorithm to extend its search in multiple directions and not to be confined to areas with lower utility levels.

$$X_i^{new} = \begin{cases} X_i + rand().(X_{Food} - I.X_i) & \text{if } f(X_i) > f(X_{FOOD}) \\ X_i + rand().(X_i - X_{FOOD}) & \text{otherwise} \end{cases} \quad (4)$$

Here, X_i^{new} is the updated position of an individual, X_{FOOD} is the location of a random individual considered as "food," and I is a random binary value.

The exploitation phase in Equation (5) refines the candidate solution by adjusting the position slightly in a way that reduces over time. Initially, to search the space of the search space, the changes made are more significant, whereas as the algorithm proceeds, the changes made are comparatively smaller, trying to get as close to the optimal value as possible. This makes sure that the algorithm is able to converge with the right solution within a short span of time.

$$X_i^{new} = X_i + 0.2.\left(1 - \frac{t}{Max\_iterations}\right).(2.rand() - 1).X_i \quad (5)$$

This equation introduces a small random perturbation to X_i as the algorithm iteratively refines the solution towards the optimal value.

4.2. Statistical Validation

The Wilcoxon rank-sum test was carried out with a significance threshold of 0.05 to provide statistical evidence that POA-DE is better than other evaluation methods. The findings indicate that POA-DE performs much better than the metaheuristics that were chosen for comparison in the majority of benchmark functions. This substantiates the fact that the observed increases in performance are statistically significant and are not the product of random chance.

5. EXPERIMENTAL SETUP AND RESULTS

A novel POA-DE method is provided in this study, followed by a comprehensive empirical analysis using 23 benchmark functions. This assessment is part of the comprehensive experimental design used to compare the effectiveness of the hybrid technique. The 23 benchmark functions are a standard set of test problems that can be used to see how the suggested changes to the POA-DE algorithm stack up against the original POA method. The test results are very important for supporting the idea that using both DE and

POA together makes the balance between exploration and exploitation better, as well as the optimization performance in many situations. The below tables and figures provide a performance comparison between the original POA and the hybrid POA-DE. The results are calculated using two parameters: The number of SA and the maximum number of iterations (MI). In cases where POA-DE performs better than POA, significant improvements (SIs) are indicated. In addition to the highest scores, each table displays the benchmark functions on which both algorithms have been evaluated. The tables are accompanied by charts that facilitate comparison. Table 2 displays the results of the POA versus POA-DE comparison for 20 SA. This table presents a comparison between the results of the POA and the outcomes of the POA-DE in situations involving 20 SA. The findings include a range of iteration numbers. They are calculating the square values of 100, 500, 800, and 1000. In addition, for each iteration count, the top results for each algorithm are shown for each benchmark function, highlighting significant improvements where POA-DE outperforms POA. To be more precise: Out of 100 trials, there are 17 instances where the performance of POA-DE exceeds that of POA in terms of SI. In 500 iterations, there are 16 improvements, with POA-DE surpassing POA by a substantial margin. Among the 800 iterations, there are 15 occurrences when the POA-DE has a superior index (SI) in comparison to the POA. Among the 1000 repetitions, there are 14 instances of SI. These findings allow for a comparison of the relative efficiency of the studied algorithms in terms of their ability to operate with a smaller group of persons. They also provide an evaluation of how many times the algorithm hybridization based on the POA improves its performance compared to the original version.

Table 3 presents the results of the comparison between the original POA and the hybrid POA-DE for scenarios with 30 SA. It includes results for various iteration counts: 100, 500, 800, and 1000. For each iteration count, the table shows the best scores achieved by each algorithm for each benchmark function and indicates significant improvements where POA-DE outperforms POA. Specifically: For 100 iterations, there are 17 significant improvements where POA-DE performs better than POA. For 500 iterations, there are 14 significant improvements where POA-DE performs better than POA. For 800 iterations, there are 14 significant improvements where POA-DE performs better than POA.

TABLE 2: POA versus POA-DE results for 20 SA

Function	Maximum number of iterations							
	100		500		800		1000	
	POA	POA-DE	POA	POA-DE	POA	POA-DE	POA	POA-DE
F1	8.44242E-19	2.607E-19	5.3111E-103	1.0405E-128	5.6066E-180	2.5394E-212	6.0024E-232	2.9654E-259
F2	1.02422E-08	1.67645E-11	1.602E-54	2.75195E-68	3.89621E-86	1.906E-111	1.2771E-111	5.7266E-147
F3	4.8004E-18	1.34672E-20	1.0435E-114	1.2884E-119	3.8238E-155	8.9934E-187	3.6462E-226	1.5807E-240
F4	1.83678E-10	4.41394E-10	3.90111E-52	5.80852E-62	6.50224E-87	3.9976E-100	1.8618E-106	3.6818E-127
F5	28.8410256	28.40746944	28.83434241	25.72787702	26.46808462	24.36385211	27.96069942	24.13425128
F6	0	0	0	0	0	0	0	0
F7	0.001842118	0.002489956	0.000811606	0.0002196	0.00018826	0.000251286	0.000195164	0.000185465
F8	−6752.338401	−8973.535957	−7986.181885	−8658.249743	−7206.595666	−9666.603987	−7690.545649	−9739.742638
F9	0	0	0	0	0	0	0	0
F10	7.1642E-10	2.5695E-12	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15
F11	0	0	0	0	0	0	0	0
F12	0.379335727	0.110466078	0.20717646	0.006759976	0.163103261	0.011113309	0.307773401	0.004877733
F13	2.993078621	1.886155787	2.485823119	0.828283394	2.974257347	0.722645709	1.996425926	0.593715012
F14	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F15	0.000308452	0.001223173	0.000307486	0.000307486	0.000307487	0.000307486	0.000307505	0.020363339
F16	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F17	0.397887372	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358
F18	3	3	3	3	3	3	3	3
F19	−3.862782044	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148
F20	−3.321732438	−3.321995121	−3.321990444	−3.20310205	−3.32199453	−3.321995172	−3.321994009	−3.321995172
F21	−10.14786467	−10.15319968	−10.15307188	−10.15319968	−10.15313299	−10.15319968	−5.055197391	−10.15319968
F22	−10.39811953	−10.40294057	−10.40283234	−10.40294057	−10.402923	−10.40294057	−10.40289467	−10.40294057
F23	−10.45228277	−10.53640982	−10.53628678	−10.53640982	−10.53639666	−10.53640982	−10.53638089	−10.53640982
SI	17		16		15		14	

POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent

TABLE 3: POA versus POA-DE results for 30 SA

Function	Maximum number of iterations							
	100		500		100		1000	
	POA	POA-DE	POA	POA-DE	POA	POA-DE	POA	POA-DE
F1	2.44686E-21	3.24652E-21	1.8596E-100	2.8411E-129	7.1447E-166	6.2389E-202	1.0337E-225	1.689E-261
F2	6.30687E-10	6.92141E-13	5.36129E-52	6.45989E-69	3.11624E-86	6.1841E-110	3.3754E-110	6.1431E-133
F3	1.49479E-18	2.15422E-19	2.1028E-111	3.8733E-127	1.5153E-172	2.2947E-192	4.3141E-216	6.8049E-241
F4	6.17297E-09	1.05369E-08	2.94986E-52	6.17893E-63	6.5319E-91	5.0517E-102	7.6164E-113	2.1422E-122
F5	28.9155397	28.5032141	28.83833557	24.70437222	27.97057944	24.06465201	25.859068	25.00836819
F6	0	0	0	0	0	0	0	0
F7	0.001223655	0.001004973	0.000328992	6.25394E-05	0.000140125	0.00022994	0.000429553	0.000181634
F8	−6427.415167	−8416.720434	−7084.244045	−8835.40723	−8185.882717	−9316.476719	−8513.675101	−8658.286337
F9	0	0	0	0	0	0	0	0
F10	8.89046E-10	2.33826E-10	4.44089E-15	8.88178E-16	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15
F11	0	0	0	0	0	0	0	0
F12	0.57331789	0.056721115	0.176276713	1.57179E-06	0.145433174	1.16027E-06	0.1679032	3.03096E-07
F13	2.990963261	1.009436307	2.620938032	0.297529501	2.982695256	0.108381791	2.980186413	0.308401823
F14	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F15	0.000308878	0.000307486	0.000307486	0.000307486	0.000307487	0.000307486	0.000307486	0.000307486
F16	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F17	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358
F18	3	3	3	3	3	3	3	3
F19	−3.862782087	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148
F20	−3.321708176	−3.20310205	−3.321993603	−3.321995172	−3.321994662	−3.20310205	−3.321991814	−3.20310205
F21	−5.055164449	−10.15319968	−10.1531993	−5.055197729	−10.15319967	−10.15319968	−10.15315661	−10.15319968
F22	−10.40167737	−10.40294057	−10.40292935	−10.40294057	−10.40281433	−10.40294057	−10.40294057	−10.40294057
F23	−10.53138633	−10.53640982	−10.5363856	−10.53640982	−10.53624667	−10.53640982	−5.128480787	−10.53640982
SI	17		14		14		15	

POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent

For 1000 iterations, there are 15 significant improvements where POA-DE performs better than POA. These results offer a comparative view of the algorithms' performance with a medium-sized population and highlight how the iteration count influences the effectiveness of the hybrid approach over the original POA.

Furthermore, Table 4 presents the results of the comparison between the original POA and the hybrid POA-DE for scenarios with 50 SA. It includes results for various iteration counts: 100, 500, 800, and 1000. For each iteration count, the table shows the best scores achieved by each algorithm for each benchmark function and indicates significant improvements where POA-DE outperforms POA. Specifically: For 100 iterations, there are 16 SI where POA-DE performs better than POA. For 500 iterations, there are 14 SI where POA-DE performs better than POA. For 800 iterations, there are 16 SI where POA-DE performs better than POA. For 1000 iterations, there are 13 SI where POA-DE performs better than POA. This table provides insights into how a larger population size affects the performance improvements of POA-DE relative to POA and the impact of iteration count on optimization results.

In last table, Table 5 presents the results of the comparison between the original POA and the hybrid POA-DE for scenarios with 80 SA. It includes results for various iteration counts: 100, 500, 800, and 1000. For each iteration count, the table shows the best scores achieved by each algorithm for each benchmark function and indicates significant improvements where POA-DE outperforms POA. Specifically: For 100 iterations, there are 17 SI where POA-DE performs better than POA. For 500 iterations, there are 13 SI where POA-DE performs better than POA. For 800 iterations, there are 13 SI where POA-DE performs better than POA. For 1000 iterations, there are 14 SI where POA-DE performs better than POA. These results reflect the performance of the algorithms with a larger population and provide a comparison of how different iteration counts affect the relative success of the hybrid POA-DE approach over the original POA.

The graphical comparisons between the POA and the hybrid POA-DE, with a focus on significant improvements, are illustrated in four figures: These are Figs. 1-4. For either figure, in this research report the results of a total of 23 test functions for given values of SAs and four iteration counts of

TABLE 4: POA versus POA-DE results for 50 SA

Function	Maximum number of iterations							
	100		500		100		1000	
	POA	POA-DE	POA	POA-DE	POA	POA-DE	POA	POA-DE
F1	2.49576E-15	8.25975E-24	6.4535E-113	1.7909E-129	1.5327E-175	3.1429E-201	2.788E-225	2.7614E-260
F2	3.883E-12	5.58281E-11	2.10498E-56	4.55152E-62	2.78613E-91	2.0983E-105	4.5724E-118	7.365E-131
F3	3.82949E-18	2.40486E-17	1.1427E-106	7.8247E-121	1.6074E-168	7.2797E-191	2.1796E-216	3.4427E-243
F4	4.78844E-09	2.42029E-10	4.97603E-51	3.37586E-58	1.05869E-85	5.01127E-98	1.4792E-106	8.7552E-127
F5	28.70319141	28.41110891	28.02387665	24.66922068	28.81369914	23.19309231	28.55704875	23.64091015
F6	0	0	0	0	0	0	0	0
F7	0.000272223	0.001243581	0.000168009	0.000191357	2.52893E-05	2.42018E-05	4.91583E-05	0.000110357
F8	−6585.887624	−6602.34042	−8065.980309	−9206.070817	−8220.373482	−9350.304939	−6827.237998	−10094.43083
F9	0	0	0	0	0	0	0	0
F10	1.02749E-10	8.60156E-11	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15
F11	0	0	0	0	0	0	0	0
F12	0.395989404	0.008500108	0.114831156	7.42445E-07	0.223673566	4.61258E-10	0.105689638	3.94851E-08
F13	2.995377503	0.195516639	2.979128781	0.097371549	2.976471966	0.010992274	2.9762112	0.240193412
F14	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F15	0.000516476	0.000307486	0.000307486	0.000307486	0.000307486	0.000307486	0.000307486	0.001223173
F16	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F17	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358
F18	3	3	3	3	3	3	3	3
F19	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148
F20	−3.321263723	−3.20310205	−3.321989525	−3.20310205	−3.321994373	−3.321995172	−3.203100353	−3.20310205
F21	−10.14426658	−10.15319968	−5.055197729	−10.15319968	−10.15319967	−10.15319968	−10.15319968	−10.315319968
F22	−10.38733962	−10.40294057	−10.40294057	−10.40294057	−10.40294057	−10.40294057	−10.40294057	−10.40294057
F23	−10.53576953	−10.53640982	−10.53640982	−10.53640982	−10.53640982	−10.53640982	−10.53640284	−10.53640982
SI	16		14		16		13	

POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent

TABLE 5: POA versus POA-DE results for 80 SA

Function	Maximum number of iterations							
	100		500		100		1000	
	POA	POA-DE	POA	POA-DE	POA	POA-DE	POA	POA-DE
F1	1.10087E-18	6.35165E-21	1.5474E-110	1.4112E-124	3.5709E-176	4.7273E-203	3.2398E-224	1.3994E-255
F2	2.66845E-10	2.18443E-10	4.80324E-53	2.51132E-60	2.11834E-89	6.4643E-109	4.9799E-112	8.4084E-130
F3	7.89952E-18	7.25637E-21	1.4617E-114	1.0093E-123	1.458E-174	2.241E-189	3.9415E-228	1.6495E-237
F4	4.06298E-09	1.62514E-10	1.68652E-54	2.84401E-58	2.39672E-84	1.54687E-98	3.4455E-106	1.2882E-119
F5	28.78665385	27.75183764	27.96056999	23.72066394	28.58014219	23.35552802	28.80080497	22.5407497
F6	0	0	0	0	0	0	0	0
F7	0.000183214	0.00046826	5.21957E-05	0.000267952	1.93001E-05	4.18375E-05	4.34409E-05	2.89613E-06
F8	−6611.55237	−7473.179641	−7089.851035	−9527.994796	−8402.996499	−10631.92435	−7543.986097	−10393.55237
F9	0	0	0	0	0	0	0	0
F10	1.03606E-11	4.31521E-11	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15	4.44089E-15
F11	0	0	0	0	0	0	0	0
F12	0.434235978	0.009326528	0.186761494	8.6929E-11	0.134381986	2.3543E-12	0.204381175	1.25909E-20
F13	2.608189819	0.062913512	1.538478871	4.31207E-10	2.172645757	6.71697E-12	2.975194915	0.09737116
F14	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838	0.998003838
F15	0.001275838	0.000307486	0.000307486	0.000307486	0.000307486	0.000307486	0.001223173	0.000307486
F16	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453	−1.031628453
F17	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358	0.397887358
F18	3	3	3	3	3	3	3	3
F19	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148	−3.862782148
F20	−3.321919745	−3.321995172	−3.321994915	−3.321995172	−3.321994196	−3.321995172	−3.32199447	−3.20310205
F21	−10.152767	−10.15319968	−10.15319965	−10.15319968	−5.055197729	−10.15319968	−10.15319968	−10.15319968
F22	−10.39964661	−10.40294057	−10.40294057	−10.40294057	−10.40293405	−10.40294057	−10.40294057	−10.40294057
F23	−10.53569662	−10.53640982	−10.5364098	−10.53640982	−10.53640982	−10.53640982	−10.53640982	−10.53640982
SI	17		13		13		14	

POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent

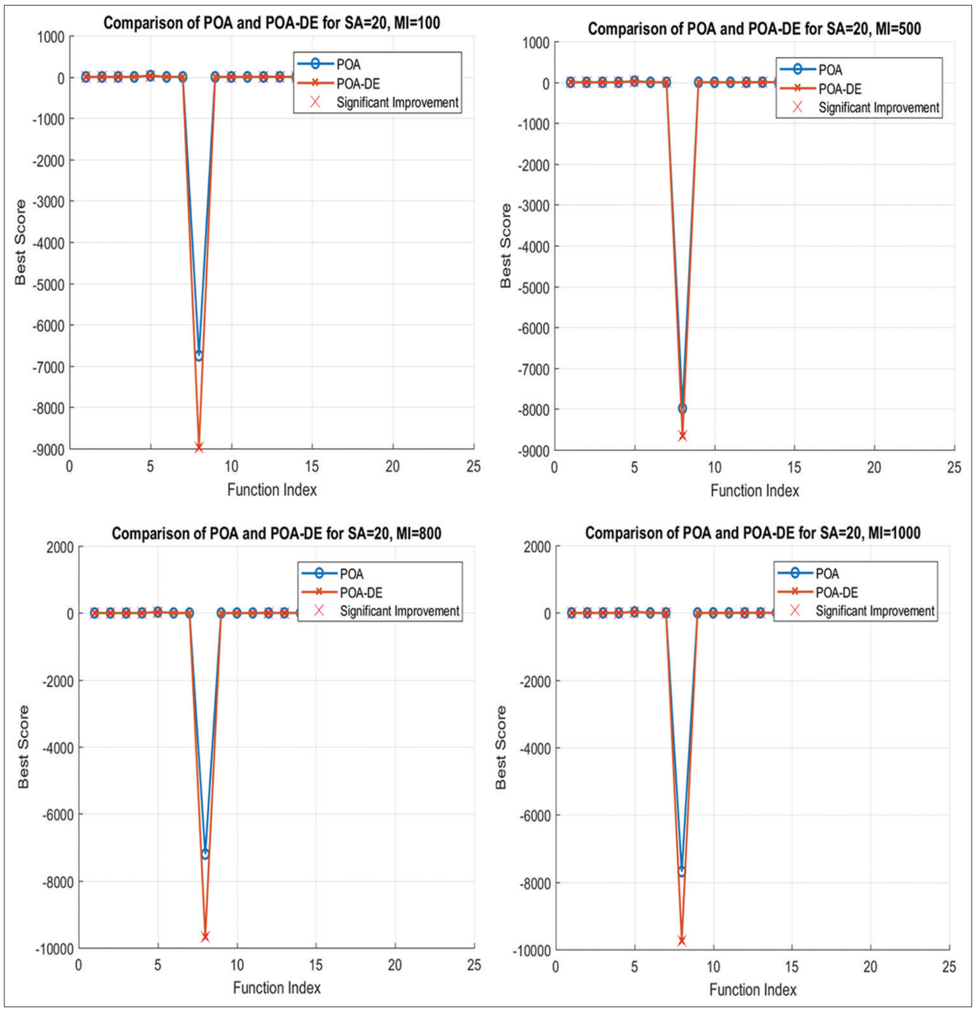

Fig. 1. Significant improvements in optimization performance: POA versus POA-DE for 20 SA. POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent.

100, 500, 800, and 1000. In all the cases, as shown in Fig. 1, when SA is set at 20, POA-DE has superior performance to POA across the board, and the differences are enhanced with more iterations. The two regions circled in red in the figure represent such enhancements; POA-DE's superior optimization results over the POA mechanism are indicated here. The same observation can be made when comparing the results presented in Fig. 2, where the results for SA = 30 are shown, which points to the fact that, in most of the test function cases, as well as at higher iteration numbers, POA-DE outperforms POA. This is the reason why the specific substantial enhancements pointed out by me underscore the efficiency of the studied hybrid design with a moderately larger quantity of populace. Figs. 3 and 4 sustain this emphasis on tremendous enhancements for SA values of 50 and 80, respectively. As indicated by the red crosses, the number of significant improvements starts to appear more frequently as the iteration count increases. This implies that with a large population size, the hybrid algorithm has more ability to search and optimally solve complex functions. The hybrid algorithm outperforms all the other algorithms in each of the settings, and the largest SA of 80 yields significant improvements from the POA to the POA-DE, as seen in Fig. 4. These figures combined show that POA-DE has a substantial improvement over the original POA, although the number of significant improvements increases with population size and iteration count, effectively substantiating the use of the hybrid model in the different test functions.

6. REAL-WORLD APPLICATION

To evaluate the real-world effectiveness of the proposed POA-DE algorithm, it was applied to practical engineering

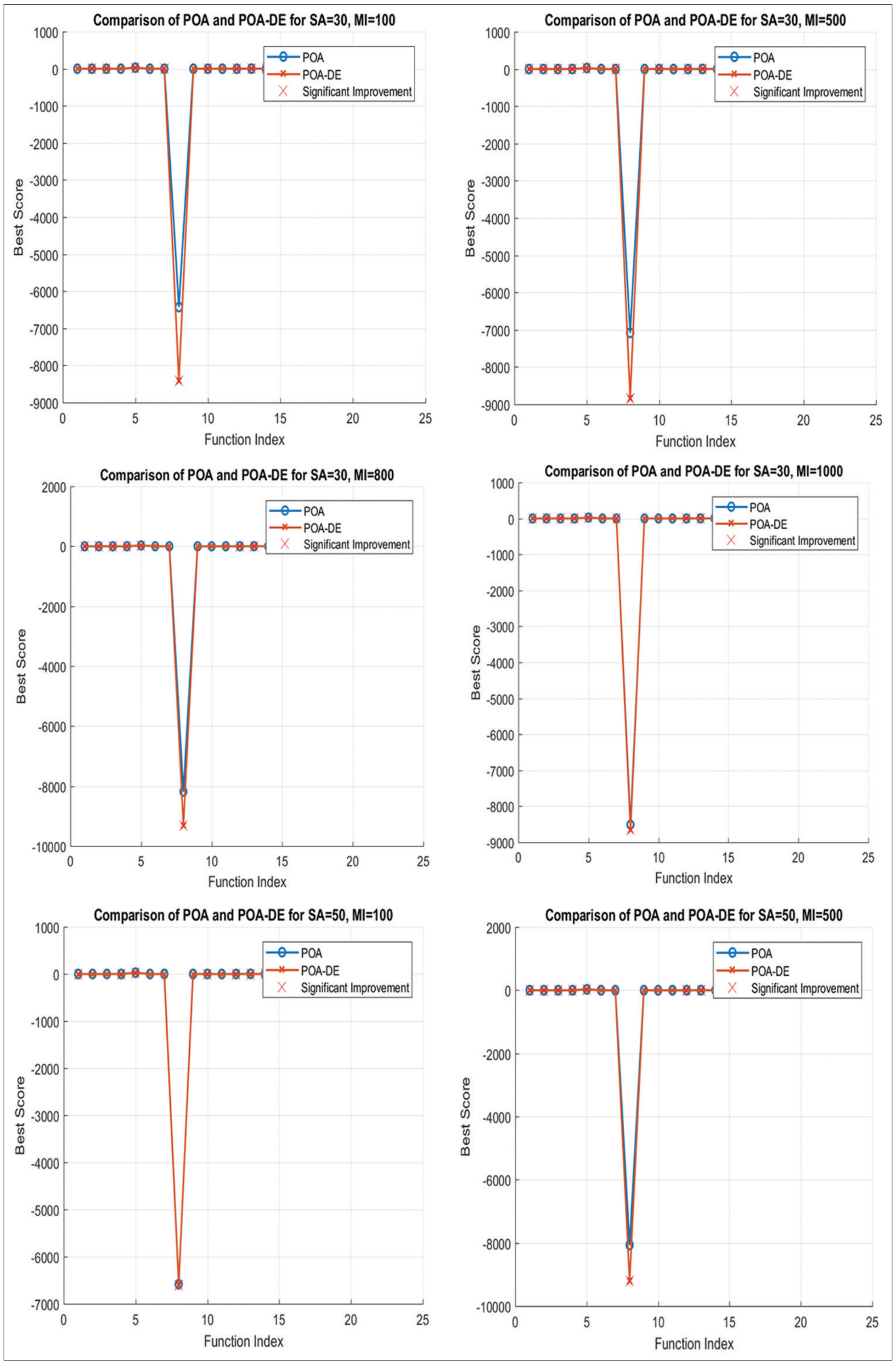

Fig. 2. Significant improvements in optimization performance: POA versus POA-DE for 30 SA. POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent.

and healthcare problems. These applications show that POA-DE works better and finds solutions faster than the original POA and other similar methods, especially in complex situations with many variables and limitations.

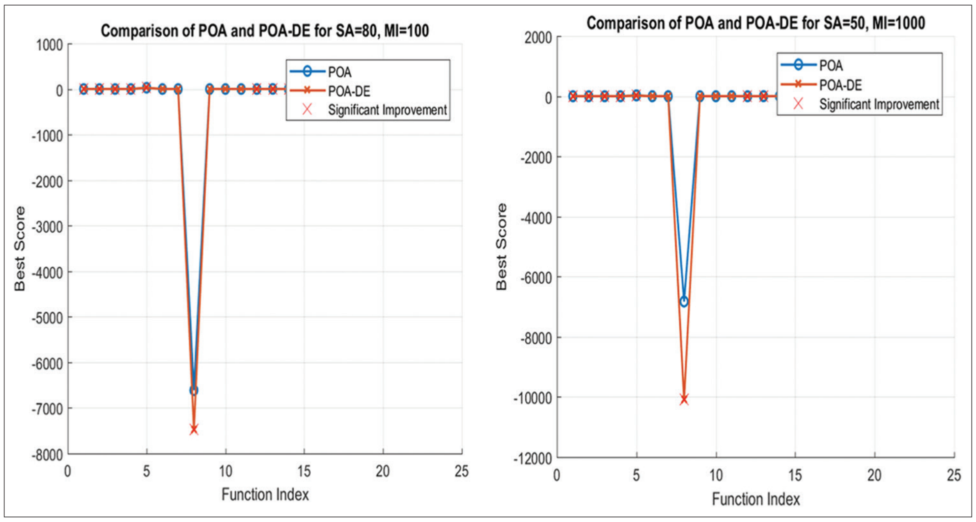

Fig. 3. Significant improvements in optimization performance: POA versus POA-DE for 50 SA. POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent.

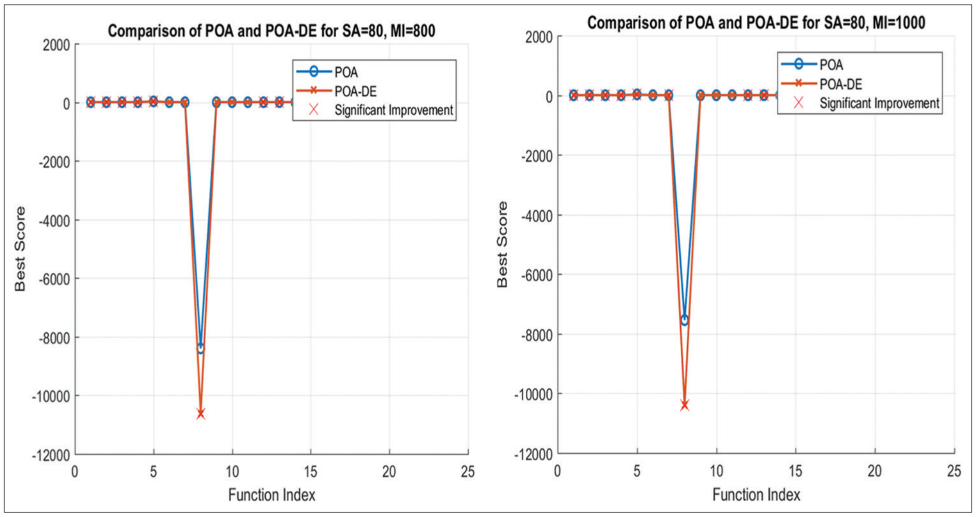

Fig. 4. Significant improvements in optimization performance: POA versus POA-DE for 80 SA. POA: Pelican optimization algorithm, DE: Differential evolution, SA: Search agent.

6.1. Medical Diagnosis–Feature Selection for Cancer Classification

POA-DE was employed to optimize feature selection for the Wisconsin breast cancer diagnostic dataset. This dataset involves diagnosing tumors as benign or malignant using 30 real-valued features. Feature selection was modeled as a binary optimization problem, aiming to maximize classification accuracy and minimize the number of selected features. A support vector machine served as the classifier, with 10-fold cross-validation to evaluate performance, as shown in Table 6.

The results show that POA-DE not only improved classification accuracy by ~1.7% over POA but also achieved it using fewer features, which reduces computational burden and enhances model interpretability. Its robust search capability in high-dimensional and noisy medical data makes it a promising candidate for clinical decision support systems.

6.2. Renewable Energy–Optimal Placement of Wind Turbines

POA-DE was applied to optimize the layout of wind turbines in a wind farm to maximize energy output while considering wake effect losses, turbine spacing constraints, and land area limitations. The wind flow was simulated based on prevailing direction and velocity using a simplified Jensen wake model, as shown in Table 7.

TABLE 6. Comparison of POA-DE and other algorithms for feature selection in breast cancer classification

Algorithm	Accuracy (%)	Selected features	Standard deviation
POA-DE	98.6	7	0.41
POA	96.9	9	0.57
GA	94.5	11	0.81
PSO	95.8	10	0.66

POA: Pelican optimization algorithm, DE: Differential evolution, PSO: Particle swarm optimization, GA: Genetic algorithm

TABLE 7: Performance comparison of POA-DE and other metaheuristics in wind farm layout optimization

Algorithm	Energy output (MWh/year)	Improvement (%)	Runtime (s)
POA-DE	14,275	+12.1	78.6
POA	12,735	___	84.2
DE	13,154	+3.3	92.4
PSO	13,079	+2.7	101.5

POA: Pelican optimization algorithm, DE: Differential evolution, PSO: Particle swarm optimization

Objective function:

- Maximize total annual energy production
- Penalize overlap or suboptimal spacing that increases wake losses.

The hybrid POA-DE produced layouts that generated up to 12.1% more energy compared to the original POA and also converged faster to near-optimal solutions. This application highlights POA-DE's ability to efficiently solve complex, non-linear, constrained engineering problems, contributing to the planning of sustainable renewable energy infrastructures.

7. CONCLUSION

This research presents a novel metaheuristic technique called POA-DE. This novel technique integrates advantageous aspects from both algorithms, leading to enhanced optimization performance in terms of solution quality and convergence as compared to the original POA. The results of the trials conducted on several benchmark functions demonstrate POA-DE's effectiveness in tackling complex optimization issues. The effectiveness of combining DE's exploitative capabilities with POA's exploratory qualities to enhance metaheuristic optimization approaches has been demonstrated. Future research will focus on enhancing and advancing numerous areas, including the following: initially, the algorithm's performance will be evaluated by subjecting it to a substantial collection of intricate real-world optimization problems. This will allow for the assessment of the technique's adaptability and reliability. Furthermore, diverse methodologies for configuring distinct parameters and implementing various types of adaptivity for the algorithm will be explored. One recommended area for future research is the integration of POA-DE with other metaheuristic approaches to provide more advanced and versatile optimization algorithms. Furthermore, it would be beneficial to do research on parallel and distributed computing approaches to effectively use them for optimizing large-scale algorithms and improving their performance. Future work will focus on several key directions to address current limitations. These include applying the algorithm to real-world optimization tasks to assess practical robustness, conducting parameter sensitivity analyses, exploring adaptive mechanisms, and extending the comparison with other advanced metaheuristics. In addition, incorporating statistical tests will ensure that performance improvements are significant and not due to randomness. These enhancements aim to further establish POA-DE as a reliable and scalable optimization framework.

REFERENCES

[1] N. A. Rashed, Y. H. Ali and T. A. Rashid. "Advancements in optimization: Critical analysis of evolutionary, swarm, and behavior-based algorithms". *Algorithms*, vol. 17, no. 9, p. 416, 2024.

[2] X. S. Yang and X. He. "Nature-inspired optimization algorithms in engineering: Overview and applications". *Studies in Computational Intelligence,* vol. 637, pp. 1-20, 2016.

[3] D. Das, A. S. Sadiq and S. Mirjalili. "Optimization methods: Deterministic versus stochastic". In: *Optimization Algorithms in Machine Learning*. Singapore: Springer, 2025.

[4] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk and W. Al-Atabany. "Honey badger algorithm: New metaheuristic algorithm for solving optimization problems". *Mathematics and Computers in Simulation*, vol. 192, pp. 84-110, 2022.

[5] E. H. Houssein, M. K. Saeed, G. Hu and M. M. Al-Sayed. "Metaheuristics for solving global and engineering optimization problems: Review, applications, open issues and challenges". *Archives of Computational Methods in Engineering*, vol. 31. pp. 4485-4519, 2024.

[6] K. Rajwar, K. Deep and S. Das. "An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges". *Artificial Intelligence Review*, vol. 56, pp. 13187-13257, 2023.

[7] V. Tomar, M. Bansal and P. Singh. "Metaheuristic algorithms for optimization: A brief review". *Engineering Proceedings*, vol. 59, no. 1, p. 238, 2024.

[8] I. Vale, A. Barbosa, A. Peixoto and F. Fernandes. "Solving authentic problems through engineering design". *Open Education Studies*, vol. 5, no. 1, p. 20220185, 2023.

[9] X. Yu, W. Chen and X. Zhang. "An Artificial Bee Colony Algorithm for Solving Constrained Optimization Problems". In: *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, Xi'an, China, pp. 2663-2666, 2018.

[10] K. R. Khudaiberganovich. "The concept of mathematical models of economic problems". *Miasto Przyszłości*, vol. 49, pp. 392-394, 2024.

[11] S. Alagarsamy, R. R. Subramanian, T. Shree, S. Kannan, M. Balasubramanian and V. Govindaraj. "Prediction of Lung Cancer Using Meta-heuristic Based Optimization Technique: Crow Search Technique". In: *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, Greater Noida, India, pp. 186-191. 2021.

[12] A. Kaveh and Y. Vazirinia. "Construction site layout planning problem using metaheuristic algorithms: A comparative study". *Iranian Journal of Science and Technology - Transactions of Civil Engineering*, vol. 43, no. 2, pp. 105-115, 2019.

[13] H. Salimi. "Stochastic fractal search: A powerful metaheuristic algorithm". *Knowledge-Based Systems*, vol. 75, pp. 1-18, 2015.

[14] P. Trojovský and M. Dehghani. "Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications". *Sensors (Basel)*, vol. 22, no. 3, p. 855, 2022.

[15] W. Tuerxun, C. Xu, M. Haderbieke, L. Guo and Z. Cheng. "A wind turbine fault classification model using broad learning system optimized by improved pelican optimization algorithm". *Machines*, vol. 10, no. 5, p. 407, 2022.

[16] Y. Han, F. Zeng, L. Fu and F. Zheng. "GA-PSO algorithm for microseismic source location". *Applied Sciences*, vol. 15, no. 4, p. 1841, 2025.

[17] L. Abualigah, A. Sheikhan, A. M. Ikotun, R. A. Zitar, A. R. Alsoud, I. Al-Shourbaji, A. G. Hussien and H. Jia. "Particle swarm optimization algorithm: Review and applications". In: *Metaheuristic Optimization Algorithms. Optimizers, Analysis, and Applications*. Elsevier Science, Amsterdam, Netherlands, pp. 1-14, 2024.

[18] M. Ahmadipour, M. M. Othman, R. Bo, M. S. Javadi, H. M. Ridha and M. Alrifaey. "Optimal power flow using a hybridization algorithm of arithmetic optimization and aquila optimizer". *Expert Systems with Applications*, vol. 235, p. 121212, 2024.

[19] S. Mirjalili and A. Lewis. "The whale optimization algorithm". *Advances in Engineering Software*, vol. 95, pp. 51-67, 2016.

[20] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. IEEE, Perth, Australia, pp. 1942-1948, 1995.

[21] R. V. Rao, V. J. Savsani and D. Vakharia. "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems". *Computer-Aided Design*, vol. 43, pp. 303-315, 2011.

[22] S. Mirjalili, S. M. Mirjalili and A. Lewis. "Grey wolf optimizer". *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014.

[23] K. Liu and Y. Wang, "A novel whale optimization algorithm based on population diversity strategy," *IAENG International Journal of Computer Science*, vol. 52, no. 8, 2025.

[24] S. Kaur, L. K. Awasthi, A. L. Sangal and G. Dhiman. "Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization". *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103541, 2020.

[25] A. Faramarzi, M. Heidarinejad, S. Mirjalili and A. H. Gandomi. "Marine predators algorithm: A nature-inspired metaheuristic". *Expert Systems with Applications*, vol. 152, p. 113377, 2020.

[26] D. E. Goldberg and J. H. Holland. "Genetic algorithms and machine learning". *Machine Learning*, vol. 3, pp. 95-99, 1988.

[27] L. N. De Castro and J. I. Timmis. "Artificial immune systems as a novel soft computing paradigm". *Soft Computing*, vol. 7, pp. 526-544, 2003.

[28] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. "Optimization by simulated annealing". *Science*, vol. 220, pp. 671-680, 1983.

[29] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi. "GSA: A gravitational search algorithm". *Information Sciences*, vol. 179, pp. 2232-2248, 2009.

[30] M. Dehghani, M. Mardaneh, J. M. Guerrero, O. Malik and V. Kumar. "Football game based optimization: An application to solve energy commitment problem". *International Journal of Intelligent Engineering and Systems*, vol. 13, pp. 514-523, 2020.

[31] A. Kaveh and A. Zolghadr. "A novel meta-heuristic algorithm: Tug of war optimization". *Iran University of Science and Technology*, vol. 6, pp. 469-492, 2016.

[32] A. Louchart, N. Tourment and J. Carrier. "The earliest known pelican reveals 30 million years of evolutionary stasis in beak morphology". *Journal of Ornithology*, vol. 152, no. 1, pp. 15-20, 2011.

[33] J. G. T. Anderson and S. C. Waterbirds. "Foraging behavior of the American white Pelican (*Pelecanus erythrorhyncos*) in Western Nevada". *Colonial Waterbirds*, vol. 14, no. 2, pp. 166-172, 1991.

[34] B. Zolghadr-Asli. "Differential evolution algorithm". In: *Computational Intelligence-based Optimization Algorithms*. CRC Press, United States, 2023.

[35] S. Das and P. N. Suganthan. "Differential evolution: A survey of the state-of-the-art". *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, 2011.