ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

# Harp Seal Optimization Algorithm Based on a Novel Selection–Combination Technique

**Solav J. Omar[1]*, Chnoor M. Rahman[1,2]**

*[1]Computer Department, College of Science, Charmo University, Sulaymaniyah, Iraq, [2]Department of Computer, College of Science, University of Sulaimani, Sulaymaniyah, Iraq*

## ABSTRACT

Metaheuristics are widely used to address optimization problems, but their efficacy varies considerably across different problem instances. This performance variability stems mainly from poor balance between exploration and exploitation. To address this limitation, this paper introduces the Harp seal (HaS) optimizer, inspired by the natural behaviors of HaSs. HaS consists of three procedures: novel selection-combination, migration, and pupping. The proposed work aims to efficiently solve a wider range of complex optimization problems. To validate the effectiveness of HaS, it was tested on 19 classical benchmarks, 10 CEC-2019 benchmarks, and a real-world application. The achieved results compared to well-known algorithms, including the genetic algorithm, multi-verse optimizer, and learner performance-based behavior. HaS outperformed or equaled other algorithms in 14 out of 19 classical benchmarks and 6 out of 10 CEC-2019 functions. It has shown robust capability across unimodal, multimodal, and composite benchmarks. The results demonstrate the proposed algorithm's capacity for both exploration and exploitation. Moreover, the good trade-off between exploration and exploitation enhances the ability of the algorithm in optimizing large-scale optimization problems. Furthermore, statistical analysis verified the significance of the observed improvements. Overall, HaS offers robust and superior results that have outperformed existing state-of-the-art algorithms.

**Index Terms:** Harp Seal Optimization Algorithm, Metaheuristic Optimization Algorithm, Evolutionary Algorithm, Harp Seal

## 1. INTRODUCTION

Optimization is the problem of searching for the near-optimal solution among the various possible solutions. Real-world problem-solving largely depends on optimization algorithms to provide superior solutions in scientific and industrial applications [1].

An optimization algorithm is designed to solve optimization problems. An optimization problem not only aims to find a solution but also evaluates its cost and effectiveness. Real-world global optimization problems are often high-dimensional and complex, involving multiple decisions and intricate relationships [2], [3].

In Abdel-Basset *et al.*'s study [4], metaheuristic methods are widely used for solving optimization problems and can be categorized in different ways. A recent classification divides them into metaphor-based and non-metaphor-based algorithms. Metaphor-based algorithms are inspired by natural, biological, or social phenomena, for instance, evolutionary algorithms, swarm intelligence, and chemical reactions. In contrast, non-metaphor-based algorithms, such as Tabu Search, rely on general, straightforward strategies without using analogies.

There are many optimization algorithms, but it is not possible for a single algorithm to solve all problems, since every algorithm has its own advantages and limitations [5].

Access this article online

**DOI:** 10.21928/uhdjst.v10n1y2026.pp69-78

**E-ISSN:** 2521-4217

**P-ISSN:** 2521-4209



**Corresponding author's e-mail:** Solav J. Omar, Department of Computer, College of Science, Charmo University, Sulaymaniyah, Iraq. Email: solav.jabar@chu.edu.iq

Received: 23-10-2025 Accepted: 30-11-2025 Published: 01-03-2026

This principle motivated the authors to propose their own metaheuristic method, which can solve a wider class of optimization problems. The proposed algorithm demonstrates particularly strong exploitation efficiency, as evidenced by its superior performance on unimodal benchmark functions. The objective of Harp Seal (HaS) is to develop a new optimizer that can efficiently tackle complex optimization problems. The main efforts in this article are summarized in the following four aspects:

- A novel leader–follower selection and combination mechanism, inspired by male–female pairing, uses distance-based matching and a leader bias weight to guide convergence toward high-quality solutions
- The pumping phase consists of two components: (1) An exploration stage using Lévy flight to expand the search space, and (2) An exploitation stage using small permutation operations to refine promising regions
- The migration phase preserves population diversity and strengthens the algorithm's ability to explore the global search space.

The performance of HaS in handling optimization problems was tested on nineteen classical benchmarks, the ten CEC-2019 test suite, and two real-world problems.

The rest of the paper is organized as follows: Section 2 provides a comprehensive related work. Section 3 details the inspiration behind the proposed algorithm. Section 4 presents the methodology of HaS, along with the pseudocode, flowchart, and computational complexity analysis. Section 5 presents the results and discussion, including a comparative study on benchmark test functions and real-world optimization problems. Finally, Section 6 concludes the study and outlines potential future research directions.

## 2. RELATED WORKS

In Amiri *et al.*'s [6] and Hazra *et al.*'s study [7], optimization methods may be classified from a set of complementary viewpoints. According to their goals, they are usually classified as single-objective, multi-objective, and many-objective. In terms of decision variables, they can be classified into continuous or discrete (binary) problems, and constrained or unconstrained, depending on whether variable boundaries are present. These categories provide a structured basis for the analysis and classification of optimization algorithms with respect to various operational properties.

From an inspiration point of view, optimization algorithms can be grouped into three major families: evolutionary algorithms, swarm intelligence algorithms, and immune system models [6], [7]. This research proposes a single objective, continuous metaheuristic optimization algorithm that can be used to optimize constrained problems such as the generalized assignment problem (GAP).

The genetic algorithm (GA) is a popular evolutionary optimization algorithm that simulates natural selection and genetic inheritance processes. GA uses three basic operators, which are selection, crossover, and mutation. Crossover with recombination mixes parent traits to develop offspring, and mutation introduces randomness to maintain diversity and escape local optima. GA efficiently explores complex solution spaces but can be computationally intensive and sensitive to parameter settings [8].

Learner performance-based behavior (LPB) Algorithm is a bio-inspired optimization algorithm motivated by the changes and adaptation in learning between high school graduates moving to university. It clusters the students into departments based on their grade point average (GPA) and dynamically adjusts their study patterns to improve academic results. Metacognitive principles are incorporated in the algorithm, focusing on self-awareness and learners' adaptive learning styles. LPB can balance well the tradeoff between exploration for new solutions and exploitation of promising solutions. However, LPB may struggle with larger combinatorial optimization problems [9].

Particle Swarm Optimization (PSO) is a popular swarm-based metaheuristic algorithm inspired by the social behavior of birds and fish, in which particles update their positions within a search space to achieve global optimization [10]. Over time, various enhancements have been introduced, along with continuous development through various other swarm-intelligence metaheuristic algorithms, such as Whale Optimization Algorithm [11], Cuckoo Optimization Algorithm [12], and Seagull Optimization Algorithm [13].

In 2024, the Puma Optimizer (PO) was proposed as a new hyper-heuristic that allows dynamic phase switching for exploration and exploitation. Novel exploration and exploitation operators, inspired by puma hunting behavior and adaptability, enhance global search ability and local exploitation. PO may face limitations in computational cost and scalability for extremely large or highly complex problems [14].

In 2025, the Dhole optimization algorithm (DOA) was proposed, inspired by the cooperative hunting and vocal communication behaviors of dholes. It incorporates adaptive decision processes and dynamic pack evolution. These mechanisms induce balance and adaptively change the searching strategy according to environmental information, thus avoiding premature convergence. However, it relies on adaptive decision rules and may require appropriate tuning [15].

WBMOAIS is an AIS-based algorithm inspired by the biological immune response, where in each iteration, solutions are regarded as antibodies that evolve through cloning, mutation, and suppression. However, a drawback of WBMOAIS is that it relies on parameter settings and may incur the computational overhead due to local search and diversity maintenance [16].

In general, current metaheuristics such as GA, PSO, LPB, PO, DOA, and WBMOAIS provide efficient optimization solutions but have limitations such as parameter setting sensitivity, premature convergence, and high computational cost in the case of complex or higher-dimensional problems. In contract, HaS successfully handles them by efficiently balancing exploration and exploitation to maintain population diversity, avoiding premature convergence through its leader–follower, pupping, and migration phases.

## 3. INSPIRATION OF HAS

HaSs are the most common pinnipeds in the North Atlantic, where mating occurs in the water, pairing the male with the female immediately after pups are weaned [17], [18]. Males select females based on proximity for mating, ensuring the closest pairs reproduce successfully [18]. HaS adopts a leader–follower strategy where leaders refer to male (strong solutions) are paired with nearby followers (weaker solutions) to generate offspring, combining their traits to guide convergence toward high-quality solutions.

HaS pups nurse on the ice for two weeks before learning to swim and dive for survival. HaS draws inspiration from this behavior. It uses a small permutation as the nursing phase for local solution refinement and Lévy flights to simulate the pups' swimming exercise for exploration, as adult females leave their pups behind on the ice and shift their focus to mating [19]. HaSs are extremely migratory, and they travel vast distances for food and proper habitats. Motivated by this phenomenon, the authors proposed a migration phase to improve global exploration and maintain a diversified population.

## 4. METHODOLOGY OF HAS

HaS is a nature-based metaheuristic algorithm that models the behavior of HaSs. Each seal represents a candidate optimization solution, and its position in the search space indicates the values of decision variables. The algorithm is initialized with a random population of candidates within a predetermined search space. Each individual's position is generated using a uniform random distribution within the given lower (*lb*) and upper (*ub*) bounds. The initial population of decision variables is generated using Eq (1).

$$x_{i,j} = lb_j + r \times (ub_j - lb_j)\ i = 1,2,3,\ldots N, j = 1,2,3,\ldots dim \quad (1)$$

Where *xi* signifies the position of the *ith* candidate solution, r is a randomly generated value ranging between 0 and 1. The parameters *lb* and *ub* define the lower and upper boundaries for the jth decision variable, respectively. *N* represents the total population of HaSs, whereas dim corresponds to the number of decision variables. The proposed algorithm is mainly divided into three phases:

Phase 1: (Selection-combination based on leader-follower: In HaS, individuals are selected for mating based on their distances, after the population has been sorted by fitness. The top 25% with the highest fitness are designated as leaders, while the remaining 75% are followers. The authors experimentally selected a ratio of 25% leaders and 75% followers to maintain an effective balance between exploitation by elite solutions (leaders) and exploration through population diversity (followers). in Eq (2) determines the number of leaders represented by NL in the population is determined when N represents the total population size, and term $0.25 \times N$ means 25% of the total top individuals are selected as leaders. Eq (3) has two sets that divide the population into two groups: the first one is L set the Leader set, and *xi* represents the leader individual. F refers to the Follower set, containing the remaining individuals *xi* represent the follower individual. Follower is paired with a leader based on distance, meaning that the follower closest to a given leader is selected for mate. After selecting leader-follower pairs, the next step blends characteristics from both the leader and the follower, ensuring a balanced search process. By structuring the population into leaders and followers, selecting the leader randomly after that select a follower based on distance, and mate the leader with the followers based on a specific weight (leader bias). In this study, combined solutions are weighted 0.96 for the leader and 0.04 for the follower. A leader bias parameter controls the leader's influence during offspring generation, ensuring new

individuals are more strongly guided by the best-performing solutions.

In this phase, the following equations are used:

$$NL = [0.25 \times N] \qquad L = \{xi | i = 1,2,\ldots,NL\} \qquad (2)$$

Where NL represents the top 25% of the population selected as leaders based on fitness. The leader set L contains the leaders, and N represent all population.

$$F = \{xj | j = NL + 1, Nl + 2, \ldots N\} \qquad (3)$$

F indicates the remaining 75% of the population, which adapts based on its proximity to leaders, and NL represents set of leaders.in Eq(4) randomly select the leader from NL.

$$leader_k = L_{rand}(1, NL) \qquad (4)$$

Where $k^{th}$ selected leader that will mate with follower, L the leader set, including the best NL individuals. Then, decide one of the leaders to be set at random.

Eq (5) calculates the Euclidean distance [20] between a selected leader and each follower associated with this leader, measuring how far each follower is from the leader. This helps identify the nearest follower for pairing with the leader, ensuring a balance between exploration and exploitation.

$$d_j = \sqrt{\sum_{m=1}^{M} (leader_m - follower_{j,m})^2} \qquad \forall x_j \in F \qquad (5)$$

Where $d_j$ The Euclidean distance between the selected leader $leader_m$ and the follower represented by $follower_{(j,m)}$ to show the positions of the leader and the $j^{th}$ follower in the $m^{th}$ dimension, respectively, while M denotes the total number of dimensions in the search space, respectively. This distance serves as a key criterion in identifying the most relevant follower for leader-follower interaction. Eq (6) chooses the closest follower by finding the one with the minimum distance.

$$follower_j = argmind_j xj \in F \qquad (6)$$

Where the follower with the minimum distance $d_j$ is selected. In Eq (7), a new individual is generated using a weighted combination using the leader bias parameter.

$$x_{new} = leader\ bias.leader + (1 - leader\ bias).follower \qquad (7)$$

Where $x_{new}$ is the newly generated individual obtained by a weighted combination of the leader and follower positions using the *leader bias* parameter. *leader bias* is a parameter used to control the influence each parent over the offspring generation process. So, the terms (*leader bias*. Leader) represent the influence of the leader's position, and (1−*leader bias*) reflects the contribution from the follower. A leader bias value of 0.96 indicates that the new individual is formed using 96% of the leader's position and only 4% of the follower's position. Such a high bias promotes exploitation, as the search process is heavily directed toward the known best (leader) solutions, encouraging convergence around high-quality areas in the solution space. However, the small contribution (4%) from the follower still retains a minimal exploratory component, depending on this equation, it has strong exploitation capability in HaS.

$$x_{new} = min(max(x\_new, lb), ub) \qquad (8)$$

The new individual generated in Eq (8) must remain within the specified *lb* lower and *ub* upper boundaries. This phase improves strong exploitation in HaS by guiding offspring generation towards leaders of high-quality solutions and retaining limited exploration through the follower influence.

Phase 2: (Pupping): HaS consists of two mathematical components, designed to improve the balance with other phases. The first part involves a small mutation, which multiplies the current solution by a small random factor (e.g., 0.005). This small randomization allows the algorithm to perform a fine-scale local exploration around the current solution. If the solution proves to be unpromising, it is discarded. This mechanism helps avoid premature convergence, effectively boosting the algorithm's exploitation capabilities. The mathematical representation of this step is shown in Eq (9).

$$x_{new} = x_i + 0.005 \cdot \left(1 - \frac{iter}{max\ Iter}\right) \cdot (ub - lb) \cdot (r_i * 0.5) \qquad (9)$$

$x_i$ represents the current solution, $x_{new}$ denotes the updated solution obtained by multiplying $x_i$ by a random factor (0.005), serving as a fine-tuning coefficient to control the mutation magnitude in such a way that only small controlled perturbations for accurate local exploitation and stable convergence. The ub and lb are the upper and lower bounds of the search space, respectively. *iter* is the current iteration, and *max Iter* is the maximum allowed iterations. Here, r is a random number in [0,1]; multiplying it by 0.5 reduces the mutation influence over time, randomizing the perturbation within [−0.5,0.5] to make the search more exploitative in later iterations. Subtracting 0.5 centers the

perturbation around zero, allowing symmetric forward and backward exploration. This adaptive mechanism keeps the search focused. This adaptive mechanism keeps the search focused around $x_i$, refines promising solutions, and prevents premature convergence, enhancing local search capabilities. The second part of the pupping phase applies Lévy flight every three iterations, enabling mostly short local moves with occasional long jumps to escape local optima and enhance exploration [21] raw Lévy step sizes are generated by Mantegna's algorithm represent by S given in Eq (10) and The Lévy step with scaling factor represent by $L_s$ computed as in Eq (11).

$$S = \frac{u}{|v|^{1/\beta}} \tag{10}$$

In Eq (10) variables $u$ and $v$ follow a normal distribution, while$\beta$ is the Lévy stability parameter that determines the likelihood of making long jumps [21].

$$L_s = Scale \times S \tag{11}$$

In Eq (11) the scale parameter as a small constant that influences step size in Lévy flights, thus maintaining stability and bounded exploration during search. In the present work, we fix this scaling factor to (0.05).

$$\sigma_U = \left( \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{(\beta-1)/2}} \right)^{1/\beta} \tag{12}$$

In Eq(12), the standard deviation of the Lévy flight step-length distribution is computed using the Gamma function (Γ), which scales the step size according to the Lévy stability index (β). This ensures adaptive control over the exploration step, allowing this algorithm to perform jumps for exploration.

Phase 3: (Migration): An important behavioral trait of HaSs is their extensive migration to find breeding, feeding, or molting areas. In HaS, this migratory habit is modeled as a global exploration strategy, guiding solutions toward promising regions. Applied every five iterations, it maintains population diversity, prevents stagnation, and enhances the algorithm's ability to escape local optima. Individuals randomly follow another HaS, denoted by *I*. Mathematically, the updated position of an individual $x_i^{new}$ is calculated according to Eq (13), where the update rule depends on a comparison of fitness values. Specifically, if the fitness of the current individual $f(x_i)$ is worse than that of the randomly selected individual $f(x_k)$, the new position is computed by moving towards $x_k$, scaled by a random factor r where r is a random scalar typically drawn from a uniform distribution in [0,1]. Otherwise, the individual moves away from $x_k$. Finally, the updated solution is clamped within the upper and lower bounds of the problem's search space Eq (14):

$$x_i^{new} = \begin{cases} x_i + r \cdot (x_k - I \cdot x_i), & if\ f(x_i) > f(x_k) \\ x_i + r \cdot (x_i - x_k), & else \end{cases} \tag{13}$$

$$x_i^{new} = \min\left(\max\left(x_i^{new}, lb\right), up\right) \tag{14}$$

The migration technique is a widely used operator in nature-based algorithms, such as the Walrus Optimization Algorithm, which helps simulate walruses' migratory behavior to enhance global exploration and avoid premature convergence [22], and Wild Geese Migration Optimization uses this technique [23]. Migration is an inherent behavior observed not only in animal species but also in humans, who migrate for better education, living standards, and economic status [24], In this phase, each solution can adjust its position based on the objective function, with one reference solution randomly selected. If the candidate has a better objective value, the current solution moves toward it, learning from fitter peers and exploring optimal regions; otherwise, it remains unchanged. This strategy achieves a good tradeoff between exploration and convergence. The pseudocode of HaS is presented in Algorithm 1 [24].

### 4.1. Computational Complexity

Computational complexity theory is a subfield of theoretical computer science that attempts to fully understand the essential complexity of computational problems and to classify them according to their computational hardness [25]. The computational complexity of HaS starts with population initialization and fitness evaluation O(N·dim), where N is the number of individuals, *dim* representing the number of decision variables, and T represents the total iteration. Each leader–follower selection, pupping and migration phases cost O(N·dim·T). Because all stages work in parallel with comparable complexity, the final complexity is O(N·dim·T). As a result, HaS scales efficiently with both population size, problem dimensionality, and number of iterations. In comparison, GA, multi-verse optimizer (MVO) time complexity equal to $O(N.dim(1+T))$, while LPB time complexity equal to O(T.N[dim+logN]). it means the

**Algorithm 1. Pseudo code of HaS**

Start HaS
***Initialization*** Generate random initial population N total iterations (**T**) within bounds **[lb, ub]**.
For t = 1 to T:
**Evaluate fitness** of all hooded seals.
**Sort population** in ascending order based on fitness.
**Divide population**:
    Leaders (Top 25%)
    Followers (Remaining 75%)
***Phase 1: Leader-follower combination***
For i=1 to N:
    Select a random leader and closest follower.
    Generate a new hooded seal using a weighted combination.
end
***Phase 2: Pupping***
For i=1 to number of leaders:
    If mod(t,3) == 0, apply Lévy flight. (exploration)
    Else, apply a small random permutation(exploitation)
end
***Phase 3: Seasonal movement (Global Exploration)***
For i=1 to N:
    every 5 iterations, apply migration
    Adjust position based on a random solution.
end
***Update population*** and store the best solution.
end
End HaS

computational complexity of HaS better than GA and MVO but slower than LPB.

## 5. RESULTS AND DISCUSSIONS

In this section, HaS algorithms are evaluated based on 29 benchmark functions and a real-world problem. The first 19 of them are classical benchmark functions and have become increasingly popular in the field as they were applied in many past studies [26], [27], [28], and are typically classified into three categories: unimodal, multimodal, and composite. HaS outperformed or matched other algorithms in 14 out of 19, achieving improvements of 34.5% over GA, 10.3% over MVO, and 35.5% over LPB across all test functions in the classical objective function. Optimization benchmarks simulate problems to assess algorithm performance [29]. To validate the scalability and robustness [30], HaS was also evaluated on 10 large-scale optimization problems, the

**TABLE 1: Comparison of results of the unimodal, multimodal, and composite test functions between GA, MVO, LPB, and HaS**

| Objective function | GA | MVO | LPB | HaS |
|---|---|---|---|---|
| F1 | 3.62E−03 | 2.01E−03 | 4.30E−03 | **0** |
| F2 | 5.87E−03 | 1.44E−02 | 4.30E−03 | **1.00E−04** |
| F3 | 5.68E+01 | 1.36E−02 | 3.47E+01 | **0** |
| F4 | 3.82E−01 | 3.36E−02 | 2.68E−01 | **2.00E−04** |
| F5 | 1.84E+01 | 1.45E+02 | 8.52E+00 | **8.22E+00** |
| F6 | 3.57E−03 | 2.38E−03 | 4.62E−03 | **0** |
| F7 | 2.55E−03 | **1.27E−03** | 4.92E−03 | 2.25E−02 |
| F8 | −3.70E+03 | −2.96E+03 | **−3.83E+03** | −2.71E+03 |
| F9 | 2.95E−03 | 1.20E+01 | 2.58E−03 | **0** |
| F10 | 2.79E−02 | 1.12E−01 | 2.46E−02 | **1.00E−04** |
| F11 | 7.39E−02 | 3.15E−01 | 5.56E−02 | **0** |
| F12 | 1.49E−04 | 2.09E−02 | 2.57E−06 | **0** |
| F13 | 9.59E−04 | 1.54E−03 | **3.78E−04** | 4.00E−03 |
| F14 | **0.998004** | **0.9980038** | **0.998** | 4.5397 |
| F15 | **1.92E−03** | 5.89E−03 | 1.65E−02 | 4.00E−03 |
| F16 | **−1.0316252** | **−1.03E+00** | **−1.03E+00** | **−1.03E+00** |
| F17 | **3.98E−01** | **3.98E−01** | **3.98E−01** | **3.98E−01** |
| F18 | 3.90E+00 | **3.00E+00** | **3.00E+00** | **3.00E+00** |
| F19 | −2.82E+00 | −3.862782 | **−3.86E+00** | **−3.86E+00** |

The bolded values indicate the best performance for each test function, representing the algorithm that achieved the smallest fitness value and therefore outperformed the other compared algorithms. GA: Genetic algorithm, MVO: Multi-verse optimizer, LPB: Learner performance-based behavior, HaS: Harp seal, WEP: Wormhole existence probability

CEC-2019 functions, achieving superior or equal results in 6 of 10 cases compared to competing algorithms. Performance was compared systematically against GA, MVO, and LPB. Each benchmark was run 30 times, and mean results are reported in Tables 1 and 2, the best values are illustrated in bold. Statistical significance was verified using the *t*-test, and a real-world application further demonstrated its effectiveness. For standard benchmarks, a population size is 50 within 1000 iterations; for CEC-2019, a population size is 80 within 1000 iterations. All the experiments were implemented in MATLAB R2024a on a MacBook Air (Apple M2, 8GB, macOS 14.6). The parameters setting is shown in Table 3.

### 5.1. Classical Benchmark Test Function

The efficiency of HaS is assessed on three groups of classical benchmark functions: Unimodal, multimodal, and composite functions. Benchmark functions are standardized test problems used to evaluate the performance, accuracy, and robustness of optimization algorithms under controlled conditions [31]. Unimodal functions have a single optimum, assessing accuracy and efficiency, while multimodal functions have multiple local optima, evaluating local search ability [32]. In the unimodal set (F1–F6), HaS outperforms other algorithms due to the effectiveness of its selection–combination, and permutation in pumping phases, which

**TABLE 2: Scalability analysis of the HaS**

| Function | 30 Dimension | | 50 Dimension | | 100 Dimension | |
|---|---|---|---|---|---|---|
| | Avg | Std | Avg | Std | Avg | Std |
| F1 | 6.57E−07 | 9.41E−07 | 1.94E−06 | 4.10E−06 | 3.15E−06 | 5.48E−06 |
| F2 | 4.01E−04 | 2.85E−04 | 6.17E−04 | 5.39E−04 | 1.03E−03 | 8.32E−04 |
| F3 | 3.25E−06 | 4.05E−06 | 1.02E−05 | 9.95E−06 | 4.01E−05 | 4.29E−05 |
| F4 | 2.64E−04 | 1.96E−04 | 2.34E−04 | 2.05E−04 | 3.71E−04 | 2.70E−04 |
| F5 | 2.87E+01 | 3.25E−02 | 4.85E+01 | 3.01E−02 | 9.84E+01 | 5.37E−02 |
| F6 | 2.66E−03 | 7.01E−04 | 2.20E−02 | 5.38E−03 | 3.97E−01 | 7.04E−02 |
| F7 | 3.84E−02 | 3.70E−02 | 3.55E−02 | 3.04E−02 | 2.94E−02 | 2.64E−02 |
| F8 | −7.33E+03 | 6.81E+02 | −1.22E+04 | 6.22E+02 | −2.36E+04 | 1.26E+03 |
| F9 | 7.79E−07 | 2.08E−06 | 8.29E−07 | 1.16E−06 | 2.01E−06 | 3.09E−06 |
| F10 | 1.47E−04 | 1.09E−04 | 1.43E−04 | 1.13E−04 | 1.73E−04 | 1.26E−04 |
| F11 | 1.42E−06 | 2.21E−06 | 1.15E−06 | 1.99E−06 | 1.41E−06 | 1.89E−06 |
| F12 | 3.49E−03 | 1.90E−02 | 1.05E−04 | 2.37E−05 | 2.80E−03 | 1.51E−03 |
| F13 | 8.82E−01 | 1.08E+00 | 3.20E+00 | 1.43E+00 | 4.74E−03 | 6.16E−03 |

Has: Harp seal

**TABLE 3: Parameter settings for HaS for classical benchmark functions**

| Algorithm | Parameter | Parameter value |
|---|---|---|
| HaS | Leader bias | 0.96 |
| | Population size for all algorithms | Classical benchmarks=50, CEC-2019=80 |
| GA | PC: Crossover percentage | PC=0.7 |
| | pm: Mutation percentage | pm=0.2 |
| | Gamma | gamma=0.05 |
| | mu: Mutation Rate | mu=0.02 |
| | Selection | Roulette wheel |
| MVO | WEP | Min (WEP) = 0.2 and Max (WEP) = 1 |
| LPB | Crossover rate | 2*round (0.7*population size) |
| | Mutation rate | round (0.2*population size) |
| | dp: the percentage of individuals chosen from M | 0.5 |

Has: Harp seal, GA: Genetic algorithm, MVO: Multi-verse optimizer, LPB: Learner performance-based behavior, WEP: Wormhole existence probability

provide strong exploitation capability, leading to superior results. However, in F7, MVO achieves better performance, which has more flat regions, making it harder for HaS to escape local optima. In the multimodal benchmarks (F8–F13), HaS delivers the best results in F9–F12, while LPB shows superior performance in F8 and F13. The strong performance of this algorithm in the multimodal benchmark set by Lévy flight and migration strategies for enhanced exploration. For the composite function (fixed- dimensional), in F14, other algorithms perform equally or slightly better than HaS, and GA shows superior performance in F15. In F16–F19, all algorithms achieve the global optimum. But HaS demonstrates high stability, identical results across these functions. Figure 1, which is consistent with the results reported in Table 1, while scalability is summarized in Table 2.

### 5.2. CEC-2019 Test Functions

The CEC 2019 benchmarks are a modern, complex, and high-dimensional test suite designed to evaluate the robustness and performance of optimization algorithms, unlike simpler classical benchmark functions [30]. As represented in Table 4, the HAS achieved the best result or the same in optimizing the CEC01, CEC02, CEC08, CEC03, CEC09, and CEC10 compared to other algorithms. For optimizing the CEC02, and CEC03, all algorithms [30] show similar results. GA achieved superior performance on the CEC06 function. MVO performed best on CEC04, while LPB outperformed all competitors on CEC05 and CEC07.

### 5.3. Real-world Application

In practice, metaheuristic techniques are extensively applied to solve practical problems. In this section, the HaS algorithm was employed to solve a GAP and an engineering problem to show the efficiency of the proposed algorithm.

#### *5.3.1 Problem definition GPA*

The GAP is a well-known NP-hard combinatorial optimization task [9]. It commonly arises in resource allocation. GAP is composed of the task assignment that assigns tasks to agents so that the total cost is minimized. In this paper, we cast GAP as the problem of assigning judicial cases to the justices' teams, aiming to minimize the overall time span needed to resolve all cases. Case assignment in judicial systems is a regular procedure, but very time-consuming, especially with the increase in case load. In order to obtain optimal allocation of cases, the HaS algorithm was used, which gave priority placement to justice teams

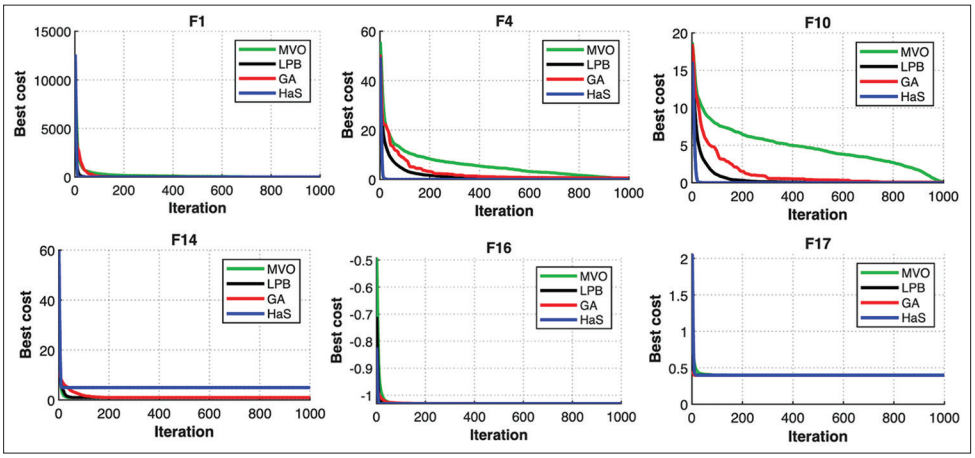


**Fig. 1.** Convergence curve for harp seal and competitor algorithm performances on unimodal, multi-modal, and composite benchmark function.

**TABLE 4: IEEE CEC 2019 benchmark test results**

| Objective function | GA | MVO | LPB | HaS |
|---|---|---|---|---|
| **CEC01** | 7.15E+04 | 1.91E+09 | 6.58E+09 | **5.27E+04** |
| **CEC02** | **1.75E+01** | **1.79E+01** | **1.79E+01** | **1.73E+01** |
| **CEC03** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** |
| **CEC04** | 4.80E+01 | **2.17E+01** | 9.78E+01 | 1.64E+02 |
| **CEC05** | 1.25E+00 | 1.24E+00 | **1.22E+00** | 2.69E+00 |
| **CEC06** | **5.40E+00** | 6.31E+00 | 5.58E+00 | 5.87E+00 |
| **CEC07** | 1.83E+02 | 2.04E+02 | **1.09E+02** | 2.83E+02 |
| **CEC08** | 5.55E+00 | 5.75E+00 | 5.35E+00 | **5.30E+00** |
| **CEC09** | 2.82E+00 | 2.40E+00 | 3.12E+00 | **2.40E+00** |
| **CEC10** | **2.00E+01** | **2.00E+01** | **2.00E+01** | **2.00E+01** |

The bolded values indicate the best performance for each test function, representing the algorithm that achieved the smallest fitness value and therefore outperformed the other compared algorithms . GA: Genetic algorithm, MVO: Multi-verse optimizer, LPB: Learner performance-based behavior, HaS: Harp seal

responsible for each case and determined the number of hours needed by them. The problem is to allocate the N cases among an amount of N justice teams, so that each team processes at most one case and minimizes the total time needed for the resolution of all cases. There is a set of constraints that mathematically formulate this problem in Eq(15). The obtained findings are compared with the results of the LPB Algorithm [9].

Mathematical formula of GPA problem: (15)

The goal is to minimize the total assignment cost, formulated as:

$$Min\, Z = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}.X_{ij}$$

Where $i$ is the row number for the $ith$ case ($i \in [1,N]$), and $j$ is the column number for the $jth$ justice team ($j \in [1,N]$). $C_{ij}$ is the cost of assigning the $ith$ case to the $jth$ justice team, and $X_{ij}$ Binary decision variable, defined as:

$X_{ij} = 1$. if case i is assigned to team j, otherwise $X_{ij} = 0$.

The double summation goes over all items and all agents, adding the cost only if that assignment is chosen $X_{ij} = 1$. Each case must be assigned to exactly one justice team.

Subject to:

$$\sum_{j=1}^{N} X_{ij} = 1 \quad , \quad \forall \epsilon\, N = \{1, 2, \ldots, N\},$$

$$\sum_{i=1}^{N} X_{ij} = 1 \quad , \forall i \in N = \{1, 2, \ldots, N\}$$

$X_{ij} \epsilon \{0,1\}$

#### 5.3.1.1 Problem representation

The problem is formulated using a row vector of dimension N, where the cost matrix is an N times N×N grid. Each element of the population is a permutation of digits between 1 and N. According to the duty assignment rule, if the $jth$ position in row is $i$, then case $i$ is assigned to justice team $j$. For instance, let's consider the following matrix:

| Case | Team 1 | Team 2 | Team 3 | Team 4 | Team 5 |
|---|---|---|---|---|---|
| Case 1 | 23 | 21 | 12 | 30 | 19 |
| Case 2 | 13 | 25 | 13 | 22 | 21 |
| Case 3 | 21 | 23 | 32 | 40 | 15 |
| Case 4 | 11 | 16 | 40 | 32 | 29 |
| Case 5 | 20 | 15 | 21 | 27 | 22 |

**TABLE 5: Comparison result for GAP**

| Size matrix | Optimal solution | |
|---|---|---|
| | HaS | LPB |
| 10×10 | 108.001 | 218 |
| 15×15 | 140.214 | 350 |
| 20×20 | 201.0201 | 425 |

GAP: Generalized assignment problem, HaS: Harp seal, LPB: Learner performance-based behavior

**TABLE 6: The *t*-test overall runs for classical benchmark test functions**

| Test functions | GA vs. HaS | MVO vs. HaS |
|---|---|---|
| F1 | **1.78E−06** | **1.19E−20** |
| F2 | **4.53E−11** | **1.83E−18** |
| F3 | **2.47E−10** | **2.02E−12** |
| F4 | 2.16E+01 | **2.87E−19** |
| F5 | **1.31E−05** | **3.10E−02** |
| F6 | **2.03E−05** | **2.09E−13** |
| F7 | **5.23E−07** | **1.96E−07** |
| F8 | **4.23E−18** | **2.08E−03** |
| F9 | **7.12E−09** | **1.13E−15** |
| F10 | **1.26E−14** | 6.47E−02 |
| F11 | **1.80E−20** | **7.40E−16** |
| F12 | **2.72E−02** | 7.69E−02 |
| F13 | 4.85E−01 | **1.08E−02** |
| F14 | 1.65E−01 | **1.05E−10** |
| F15 | **1.05E−10** | 2.31E−01 |
| F16 | 5.00E−01 | No deference |
| F17 | No deference | No deference |
| F18 | No deference | No deference |
| F19 | No deference | No deference |

The bolded values indicate the best performance for each test function, representing the algorithm that achieved the smallest fitness value and therefore outperformed the other compared algorithms.

For example, based on the cost matrix, the solution vector [2 4 1 3 5] indicates that Case 2 is assigned to Team 1 (cost = 13), Case 4 to Team 2 (cost = 16), and Case 1 to Team 3 (cost = 12), ensuring that each case is uniquely assigned to exactly one team. The problem was solved using 80 solutions over 200 iterations with HaS. The produced results were then compared to the LPB results. The results are summarized in Table 5, showing the mean of the best fitness, validating the effectiveness of HaS in solving the GAP, outperforming LPB across three different problem scales. This superior performance is attributed to HaS's balanced exploration–exploitation strategy, achieved through selection–combination, migration, and pupping, which enables both efficient global search and precise local refinement. Such a balance allows HaS to escape local optima and achieve faster and more stable convergence in complex assignment problems. Then, the convergence curve of the algorithm for solving the problem that is described by a 10×10 cost matrix is shown in Figure 2.

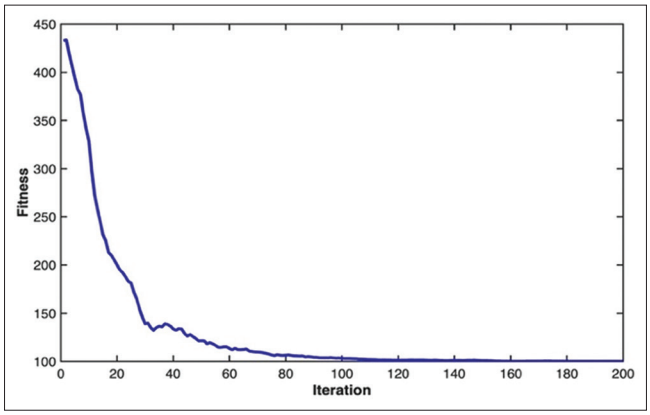


**Fig. 2.** Convergence curve for grade point average (GPA) problem size: 10×10.

### 5.4. Statistical Tests

Student's *t*-test is a parametric test that focuses on determining if there is a important difference between the means of two data samples [33]. To statistically validate the performance of HaS against GA and MVO, a *t*-test was conducted, and the corresponding *P*-values are reported in Table 6. The results reveal that HaS demonstrates statistically significant superiority ($P < 0.05$) in 12 out of 19 classical benchmark functions. For the remaining functions (F7 to F19) in comparison with GA, and (F16 to F19) with MVO, all algorithms converged to the global optimum, yielding identical results. In the table, these results are represented as no difference.

## 6. CONCLUSION AND FUTURE WORKS

In this study, a new metaheuristic algorithm named HaS is proposed based on the natural actions of HaSs, HaS was designed based on novel selection-combination, pupping (small permutation with Lévy flight), and migration mechanism. A parameter controls the contribution of leaders and followers in producing new individuals. This algorithm's performance was evaluated on traditional benchmark functions, CEC-2019 test functions, and real-world applications, including the NP-hard GAP. Comparative results with well-known algorithms show that HaS outperforms GA, MVO, and LPB on most benchmarks, demonstrating effective exploration, exploitation, and balance between them. Statistical analysis confirmed its significance. While real-world application further validated its robustness, it was not the top performer on a few benchmarks.

For future work, two directions are proposed: (i) Extending HaS to a dual-population, multi-objective constrained framework to

solve complex constrained problems. (ii) dynamically adjusting the leader-bias parameter to enhance adaptability.

## REFERENCES


[1] T. Rahkar Farshi. "Battle royale optimization algorithm". *Neural Computing and Applications*, vol. 33, no. 4, pp. 1139-1157, 2021.

[2] E. Trojovska, M. Dehghani and P. Trojovsky. "Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm". *IEEE Access*, vol. 10, pp. 49445-49473, 2022.

[3] P. Agrawal, H. F. Abutarboush, T. Ganesh and A. W. Mohamed. "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)". *IEEE Access*, vol. 9, pp. 26766-26791, 2021.

[4] M. Abdel-Basset, L. Abdel-Fatah and A. K. Sangaiah. "Metaheuristic algorithms: A comprehensive review". In: *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Amsterdam: Elsevier, 2018, pp. 185-231.

[5] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.

[6] M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili and N. Khodadadi. "Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm". *Sci Rep*, vol. 14, no. 1, pp. 5032, 2024.

[7] A. Hazra, P. Rana, M. Adhikari and T. Amgoth. "Fog computing for next-generation internet of things: Fundamental, state-of-the-art and research challenges". *Computer Science Review*, vol. 48, p. 100549, 2023.

[8] D. Whitley. "A genetic algorithm tutorial". *Statistics and Computing*, vol. 4, pp. 65-85, 1994.

[9] C. M. Rahman and T. A. Rashid. "A new evolutionary algorithm: Learner performance-based behavior algorithm". *Egypt Informatics Journal*, vol. 22, no. 2, pp. 213-223, 2021.

[10] Y. Zhang, S. Wang and G. Ji. "*A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications*". Hindawi Limited, Egypt, 2015.

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

[12] R. Rajabioun. "Cuckoo optimization algorithm". *Applied Soft Computing*, vol. 11, no. 8, pp. 5508-5518, 2011.

[13] G. Dhiman and V. Kumar. "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems". *Knowledge-Based Systems*, vol. 165, pp. 169-196, 2019.

[14] B. Abdollahzadeh, N. Khodadadi, S. Barshandeh, P. Trojovsky, F. S. Gharehchopogh, E. M. El-Kenawy, L. Abdollahzadeh and S. Mirjalili. "Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning". *Cluster Computing*, vol. 27, pp. 5235-5283, 2024.

[15] B. O. Mohammed, H. S. Aghdasi and P. Salehpour. "Dhole optimization algorithm: A new metaheuristic algorithm for solving optimization problems". *Cluster Computing*, vol. 28, p. 430, 2025.

[16] J. Gao and J. Wang. "WBMOAIS: A novel artificial immune system for multiobjective optimization". *Computers and Operations Research*, vol. 37, no. 1, pp. 50-61, 2010.

[17] G. B. Stenson, T. Haug, and M. O. Hammill. "*Harp Seals: Monitors of Change in Differing Ecosystems*". Frontiers Media S.A. Lausanne, 2020.

[18] R. F. Theiler, J. S. Siuda and L. P. Hager. "Bromoperoxidase from the red algae *Bonnemaisonia hamifera*," In: P. N. Kaul and C. J. Sindermann, eds. *Drugs and Food from the Sea: Myth or Reality*. University of Oklahoma Press, Norman, OK, USA, 1978, pp. 153-169.

[19] K. M. Kovacs, D. M. Lavigne and S. Innes. "Mass transfer efficiency between harp seal (*Phoca groenlandica*) mothers and their pups during lactation". *Journal of Zoology*, vol. 223, no. 2, pp. 213-221, 1991.

[20] L. Wang, Y. Zhang and J. Feng. "On the euclidean distance of images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1334-1339, 2005.

[21] R. N. Mantegna. "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes". *Physical Review E*, vol. 49, no. 5, pp. 4677-4683, 1994.

[22] P. Trojovský and M. Dehghani. "A new bio-inspired metaheuristic algorithm for solving optimization problems based on walruses behavior". *Scientific Reports*, vol. 13, no. 1, 2023.

[23] H. Wu, X. Zhang, L. Song, Y. Zhang, L. Gu and X. Zhao. "Wild geese migration optimization algorithm: A new meta-heuristic algorithm for solving inverse kinematics of robot". *Computational Intelligence and Neuroscience*, vol. 2022, 1-38,

[24] P. Trojovský and M. Dehghani. "Migration algorithm: A new human-based metaheuristic approach for solving optimization problems". *CMES - Computer Modeling in Engineering and Sciences*, vol. 137, no. 2, pp. 1695-1730, 2023.

[25] F. Neukart. "*Thermodynamic Perspectives On Computational Complexity: Exploring the P vs. NP Problem*". 2023. Available from: https://arxiv.org/abs/2401.08668 [Last accessed on 2025 Sep 10].

[26] M. Khishe and M. R. Mosavi. "Chimp optimization algorithm". *Expert Systems with Applications Journal*, vol. 149, 113338, 2020.

[27] J. G. Digalakis and K. G. Margaritis. "On benchmarking functions for genetic algorithms". *International Journal of Computer Mathematics*, vol. 77, no. 4, pp. 481-506, 2001.

[28] X. S. Yang. "A new metaheuristic bat-inspired algorithm". In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Studies in Computational Intelligence*. vol. 284. Berlin: Springer, 2010, pp. 65-74.

[29] Y. He and C. Aranha. "*Evolving Benchmark Functions to Compare Evolutionary Algorithms via Genetic Programming*". 2024. Available from: https://arxiv.org/abs/2403.14146 [Last accessed on 2025 Sep 02].

[30] A. A. Ameen, T. A. Rashid and S. Askar. "CDDO-HS: Child Drawing Development Optimization-Harmony Search Algorithm". *Applied Sciences*, v*ol.* 13, no. 9, p. 5795.

[31] J. M. Dieterich and B. Hartke. "*Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization*". 2012. Available from: https://arxiv.org/abs/1207.4318 [Last accessed on 2025 Jul 06].

[32] Y. Li, L. Li, Z. Lian, K. Zhou and Y. Dai. "A quasi-opposition learning and chaos local search based on walrus optimization for global optimization problems". *Scientific Reports*, vol. 15, no. 1, p. 2881, 2025.

[33] C. Wang and J. Jia. "*Te Test: A New Non-asymptotic T-Test for Behrens-Fisher Problems*". 2022. Available from: https://arxiv.org/abs/2210.16473 [Last accessed on 2025 Sep 06].