

Holistic swarm optimization: A novel metaphor-less algorithm guided by whole population information for addressing exploration-exploitation dilemma

Ebrahim Akbari ^a, Abolfazl Rahimnejad ^{b,c,*}, Stephen Andrew Gadsden ^c

^a Department of Economics, University of Bergamo, Via dei Caniana 2, 24127, Bergamo, BG, Italy

^b Department of Computer Science, Georgia Institute of Technology, North Avenue, Atlanta, GA, 30332, USA

^c Department of Mechanical Engineering, McMaster University, 1280 Main St W, Hamilton, ON, N1L1S3, Canada

ARTICLE INFO

Keywords:

Adaptive mutation
Metaphor-less optimization algorithms
Population-based algorithms
Simulated annealing-based selection
Swarm optimization

ABSTRACT

This paper introduces a novel metaphor-less optimization algorithm called Holistic Swarm Optimization (HSO), designed to enhance the search process by utilizing data from the entire population. Unlike conventional algorithms that rely on partial or local information, HSO adopts a comprehensive approach, ensuring that each decision is informed by the overall distribution and fitness landscape of the population. The algorithm dynamically balances exploration and exploitation through an adaptive framework that integrates root-mean-squared (RMS) fitness-based displacement coefficients, simulated annealing-based selection, and adaptive mutation. This structure enables HSO to efficiently navigate complex, multimodal optimization problems while avoiding local optima. The performance of HSO is evaluated on two widely used benchmark test suites—CEC 2005 and CEC 2014—and a series of real-world engineering design problems. Results show that HSO delivers competitive and stable performance when compared to several state-of-the-art metaphor-based and metaphor-less algorithms. These findings demonstrate the effectiveness of a holistic population-guided approach in achieving robust optimization outcomes, making HSO a promising alternative for solving diverse and challenging problems without reliance on metaphorical inspirations.

The source codes and implementation guidance for the HSO algorithm are available for public access on the <https://github.com/ebrahimakbari/HSO>.

1. Introduction

Optimization algorithms play a crucial role in solving complex problems across diverse scientific and engineering fields. These algorithms focus on finding the best solution by either maximizing or minimizing objective functions within the confines of specific constraints. Traditional approaches to optimization, such as deterministic and gradient-based methods, often fall short when tackling the complexities of contemporary optimization problems, which may involve vast search spaces, nonlinear relationships, and multiple local optima. In response, metaheuristic algorithms have gained prominence as flexible and adaptable alternatives, effectively addressing a broad spectrum of optimization challenges [1–4].

* Corresponding author.

E-mail address: a.rahimnejad@mcmaster.ca, a.rahimnejad@gatech.edu (A. Rahimnejad).

<https://doi.org/10.1016/j.cma.2025.118208>

Received 16 August 2024; Received in revised form 20 April 2025; Accepted 4 July 2025

Available online 17 July 2025

0045-7825/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC license (<http://creativecommons.org/licenses/by-nc/4.0/>).

1.1. Population-based metaheuristic algorithms

Population-based metaheuristic algorithms excel in navigating and optimizing complex solution landscapes through iterative processes. These algorithms progressively refine initial solutions or populations of solutions, effectively balancing between global exploration and local exploitation. Their key strengths include independence from gradient information, the ability to address multi-objective problems, and broad applicability across various fields. Nevertheless, they also face challenges such as the lack of guaranteed convergence to a global optimum, difficulties in dealing with high-dimensional spaces, and the necessity for careful tuning of control parameters [5,6].

Recent developments in the optimization literature have proposed innovative methods across a range of applications [7–11]. These recent contributions highlight the diverse and rapidly evolving nature of metaheuristic and hybrid optimization methods. In recent years, the literature on metaheuristics has grown substantially. Currently, over 600 metaheuristic algorithms exist, with more than 400 developed within the last decade alone. New optimization algorithms, often based on metaphorical concepts, are frequently introduced, with their creators claiming novelty and superiority over existing methods. The last 15 years have seen the emergence of numerous “novel” population-based metaheuristics inspired by a wide range of natural phenomena, including natural disasters, animals, birds, insects, viruses, diseases, human behaviors, celestial bodies, plants, physical laws, chemical processes, and even mathematical concepts [12].

However, the increasing reliance on metaphors has sparked concerns about the scientific robustness and practical relevance of these algorithms. Critics argue that many metaphor-based algorithms lack a strong theoretical underpinning and offer limited contributions to the advancement of optimization science. Some researchers have expressed skepticism about the surge in new algorithms, pointing out that many are built on arbitrary metaphors with little or no connection to the actual optimization process [13,14]. This trend is seen as potentially harmful to the field of optimization, as it may lead to a significant portion of the scientific community endorsing metaphor-based algorithms without substantial proof of their effectiveness [13,14].

For example, researchers in Sörensen [14] voiced concerns that the current direction of metaheuristic research risks deviating from rigorous scientific principles. Campelo et. al. compiled an extensive list of “novel” algorithms and demonstrated that developing a metaheuristic based solely on a real-world analogy is often unproductive and should not be added to the scientific literature [13]. Multiple researchers have highlighted the troubling trend of numerous metaphor-based metaheuristics, where the metaphor, the derived mathematical model, and the algorithm itself are treated as distinct entities [12].

1.2. Advancements in metaphorless optimization algorithms

In response to the growing concerns regarding metaphor-based approaches, researchers have increasingly focused on developing metaphorless optimization algorithms that prioritize mathematical rigor and simplicity. One prominent example is the Jaya optimization algorithm, proposed by Rao in 2015 [15]. Jaya is a straightforward yet powerful method for addressing both constrained and unconstrained optimization problems. Its core principle revolves around iteratively guiding candidate solutions towards the best solution while avoiding the worst, without requiring algorithm-specific control parameters, making it particularly user-friendly.

Building on this foundation, Rao introduced three non-metaphorical optimization algorithms in 2020 [16], collectively known as Rao algorithms. These methods are characterized by their reliance on the best and worst solutions in each iteration, as well as interactions among candidate solutions. Like Jaya, they are free from algorithm-specific control parameters, focusing only on population size and the number of iterations. An advanced variant of these algorithms, the Fully Informed Search Algorithm (FISA) [17], was developed in 2020 to maintain simplicity while enhancing performance in real-world optimization problems. FISA has demonstrated effectiveness in optimizing shifted benchmark functions, such as those from CEC 2005 and CEC 2014, and in the design of mechanical system components, outperforming the original Rao algorithms.

Another notable metaphor-free algorithm is the RUNge Kutta optimizer (RUN) [18], inspired by the mathematical principles of the Runge Kutta method. RUN addresses the drawbacks of metaphor-based optimizers by utilizing slope variations for global optimization through active exploration and exploitation phases. The algorithm also incorporates an enhanced solution quality (ESQ) mechanism to avoid local optima and accelerate convergence, making it a promising tool for various optimization challenges.

The max-min-greedy-interaction (MaGI) algorithm [19] represents another recent advancement in metaphorless optimization. MaGI leverages the interaction between maximum and minimum values within a candidate group to refine solution updates, showcasing superior performance in both exploration and exploitation tasks.

Additionally, two efficient metaphor-free algorithms, the Best-Mean-Random (BMR) and Best-Worst-Random (BWR) methods, have been recently proposed [12]. These approaches utilize straightforward population-based strategies without the need for algorithm-specific control parameters and have proven to be highly effective and reliable across a wide range of benchmark problems.

1.3. Proposed algorithm

Building on these advancements, this paper introduces a novel metaphorless optimization algorithm called the Holistic Swarm Optimization (HSO). Unlike traditional methods that rely on partial information or subsets of the population, HSO utilizes data from the entire population to inform its update mechanism. This comprehensive approach ensures that the algorithm's decisions are based on a holistic understanding of the population's overall distribution and fitness landscape. Consequently, this leads to a more robust and thorough search process, as decisions are grounded in extensive insights rather than isolated samples.

HSO also excels in dynamically balancing exploration and exploitation. The algorithm adjusts its strategies in real-time based on feedback, continually optimizing the balance between searching new areas (exploration) and refining known good solutions (exploitation). This dynamic balancing is crucial for avoiding local optima and ensuring the algorithm can reliably find an optimal solution from a comparatively global perspective.

The proposed HSO algorithm offers several key contributions and novelties as per following:

- **Comprehensive Population Utilization:** The HSO algorithm leverages data from the entire population to inform the update mechanism of each search agent. This comprehensive approach ensures that the algorithm's decisions are based on a holistic understanding of the population's overall distribution and fitness landscape.
- **Effective Displacement Coefficients:** The algorithm introduces a novel method for calculating displacement coefficients based on the root-mean-squared (RMS) fitness values. This method ensures that agents move towards better solutions and away from worse ones, enhancing the algorithm's convergence properties.
- **Adaptive Mutation:** HSO incorporates adaptive mutation to enhance exploration capabilities. This feature introduces random perturbations with a certain probability, helping the algorithm to explore new areas of the search space and avoid local optima.
- **Adaptive Simulated Annealing-Based Selection:** The algorithm employs simulated annealing-based selection to decide whether to accept new positions based on their fitness. This technique helps balance exploration and exploitation by accepting worse solutions with a probability that decreases over time. In fact, the probability of selection is inversely related to the level of the unfavorable change in the latest iteration and it also decreases as the iterations pass.
- **Robustness Across Domains:** The HSO algorithm has demonstrated competitive performance across various benchmark problems, proving its efficacy and reliability. Its robust framework makes it suitable for solving complex optimization problems in diverse fields.

The primary aim of this paper is not to disparage the work of researchers who have developed or are currently developing metaphor-based optimization algorithms. Instead, it seeks to establish that relying on metaphors is not a prerequisite for the creation of new optimization algorithms. Although the equations formulated in such research may successfully address benchmark functions and other challenges, the analogies drawn with natural or artificial phenomena are not fundamentally necessary. A more prudent and academically rigorous approach for researchers would be to focus on developing straightforward optimization methodologies that are capable of solving complex problems more effectively, rather than pursuing the creation of metaphor-based algorithms. Therefore, the primary objectives of this research are:

- To propose a novel metaphorless optimization algorithm that exploits the whole population information to handle the search process that is also reinforced by adaptive mutation and selection operators to balance between exploration and exploitation.
- To demonstrate the efficacy of the proposed metaphorless optimization algorithm in solving complex optimization problems.
- To compare the performance of the proposed algorithm with existing metaphorless and metaphor-based Meta-Heuristic optimization algorithms.
- To highlight the advantages of using a comprehensive population-based approach over traditional methods that rely on partial or random sampling of the population.

Compared to prior optimization algorithms, our proposed Holistic Swarm Optimization (HSO) framework introduces several key distinctions. Unlike traditional metaphor-based algorithms that draw analogies from physical, biological, or social phenomena, HSO is entirely metaphor-free and driven by population-level mathematical constructs. While existing metaphor-less algorithms such as JAYA [15] and the Rao family [16] utilize best-worst solution strategies or pairwise interactions, HSO uniquely exploits whole-population feedback using root-mean-squared (RMS) fitness values to define displacement coefficients for each agent. This approach results in an informed and directional search across the entire swarm. Furthermore, unlike other parameter-free methods, HSO integrates adaptive mutation and simulated annealing-based selection in a dynamic scheme to regulate exploration and exploitation. This hybrid structure allows the algorithm to avoid premature convergence more effectively and ensures robustness across diverse problem landscapes. These distinctions enable HSO to outperform both traditional metaphor-based and recently proposed metaphor-less optimizers on standard benchmarks and real-world problems.

The remainder of this paper is structured as follows: [Section 2](#) presents the detailed formulation and steps of the Holistic Swarm Optimization (HSO) algorithm, including population initialization, fitness evaluation, coefficient calculation, position update, adaptive simulated annealing-based selection, and adaptive mutation. [Section 3](#) evaluates the computational performance of HSO on benchmark functions, specifically focusing on the CEC 2005 and CEC 2014 benchmark suites. A discussion on the exploration-exploitation capability of HSO, using population diversity as a metric, is also included in this section. [Section 4](#) showcases the numerical results of HSO applied to various engineering optimization problems, such as the optimal design of a pressure vessel, tension/compression spring, cantilever beam, three-bar truss, and welded beam, as well as the optimization of a speed reducer design. Finally, [Section 5](#) concludes the paper by summarizing the findings and discussing potential avenues for future research.

2. Holistic Swarm Optimization (HSO)

The Holistic Swarm Optimization (HSO) algorithm is designed to leverage the entire population's information to update each member's position comprehensively. By utilizing data from all agents, HSO ensures that the search process is informed by a complete understanding of the population's overall distribution and fitness landscape. This approach leads to a more robust and thorough search,

dynamically balancing exploration and exploitation based on real-time feedback. The different phases of the proposed algorithm are introduced in the following sub-sections. The main phases of the proposed HSO algorithm are listed and discussed as follows:

2.1. Population initialization

The algorithm begins by initializing a population of search agents, denoted as $\mathbf{x}_i$ for $i = 1, 2, \dots, n$, where n is the number of agents. Each agent's position is randomly generated within the defined search space. Each search agent $\mathbf{x}_i$ represents a potential solution to the optimization problem, which is m -dimensional itself.

2.2. Fitness evaluation

The fitness of each search agent is evaluated based on the objective function $f(\mathbf{x}_i)$.

2.3. Coefficient calculation

The root-mean-squared (RMS) of all fitness values, denoted as RMS_f , is calculated as follows:

$$\text{RMS}_f = \sqrt{\frac{1}{n} \sum_{i=1}^n f(\mathbf{x}_i)^2} \quad (1)$$

where $f(\mathbf{x}_i)$ is the fitness value of agent i . Then, the coefficients for displacement vectors are computed based on the difference between each fitness value and the RMS. The difference for agent i is given by:

$$d_i = \text{RMS}_f - f(\mathbf{x}_i) \quad (2)$$

The coefficients c_i are assigned positive values for differences below the RMS and negative for those above, normalized to sum up to 1 in absolute terms:

$$c_i = \frac{\text{sign}(d_i) \cdot |d_i|}{\sum_{j=1}^n |d_j|} \quad (3)$$

where $\text{sign}(d_i)$ is the sign function, indicating the direction of movement for each agent.

2.4. Position update

The positions of the search agents are updated iteratively. Each agent $\mathbf{x}_i$'s new position $\mathbf{x}_i'$ is influenced by the weighted sum of the differences between its current position and the positions of all other agents. The update rule is:

$$\mathbf{x}_i' = \mathbf{x}_i + \alpha \sum_{j=1}^n r_{ij} \cdot c_j \cdot (\mathbf{x}_j - \mathbf{x}_i) \quad (4)$$

where α is a constant parameter, and r_{ij} is a random value that ensures variability and comprehensive exploration of the search space.

2.5. Adaptive simulated annealing-based selection

The simulated annealing (SA)-based selection process includes a dynamic temperature update at each iteration. The temperature T is updated as follows:

$$T = T_{\text{initial}} \times \text{CR}^{\text{iter}} \quad (5)$$

where T_{initial} is the initial temperature, CR (Cooling Rate) is a parameter that controls the rate of temperature decrease, and iter is the current iteration number.

If new fitness value $f_{\text{new}}(\mathbf{x}_i)$ is better than the previous fitness value $f(\mathbf{x}_i)$, the algorithm operates in a greedy manner and replaces the old solution with the newer one. On the other hand, if the new fitness value $f_{\text{new}}(\mathbf{x}_i)$ is worse than the previous fitness value $f(\mathbf{x}_i)$, the following criteria are applied for selection:

$$\Delta f = f_{\text{new}}(\mathbf{x}_i) - f(\mathbf{x}_i) \quad (6)$$

The new position is accepted with a probability P given by:

$$P = \exp\left(-\frac{\Delta f}{T}\right) \quad (7)$$

If P is greater than a random number drawn from a uniform distribution $\text{Unif}_{\text{rand}}$, the new position is accepted despite the decrease in fitness:

$$\text{if } \exp\left(-\frac{\Delta f}{T}\right) > \text{Unif}_{\text{rand}} \quad (8)$$

In summary, according to the SA-based selection operator, the probability of selection of a specific solution that reaches a position worse than its previous position in a certain iteration is inversely proportional to the level of the unfavorable change in its fitness value in the most recent iteration, and this probability becomes less and less as the iteration proceeds. This setup balances exploration and exploitation over time.

2.6. Adaptive mutation

To further enhance exploration capabilities, a mutation operator is defined in an adaptive manner so that its mutation rate and mutation step size are dynamically adjusted based on the current iteration. The mutation rate $\lambda_{\text{mut}}^{\text{iter}}$ and mutation step size $\delta_{\text{mut}}^{\text{iter}}$ are calculated as follows:

$$\lambda_{\text{mut}}^{\text{iter}} = \lambda_{\text{init}} - \text{iter} \times \left(\frac{\lambda_{\text{init}} - \lambda_{\text{final}}}{\text{iter}_{\text{max}}} \right) \quad (9)$$

where $\lambda_{\text{mut}}^{\text{iter}}$ is the mutation rate at iteration iter , λ_{init} is the initial mutation rate, λ_{final} is the final mutation rate, iter is the current iteration, and iter_{max} is the maximum number of iterations.

$$\delta_{\text{mut}}^{\text{iter}} = \delta_{\text{init}} - \text{iter} \times \left(\frac{\delta_{\text{init}} - \delta_{\text{final}}}{\text{iter}_{\text{max}}} \right) \quad (10)$$

where $\delta_{\text{mut}}^{\text{iter}}$ is the mutation step size at iteration iter , δ_{init} is the initial mutation step size, and δ_{final} is the final mutation step size.

The mutation rate $\lambda_{\text{mut}}^{\text{iter}}$ dictates the probability of a mutation occurring during iteration iter . As the number of iterations increases, $\lambda_{\text{mut}}^{\text{iter}}$ decreases linearly from its initial value λ_{init} to its final value λ_{final} . Similarly, the mutation step size $\delta_{\text{mut}}^{\text{iter}}$ controls the magnitude of the perturbation applied to the search agent's position. It also decreases linearly from its initial value δ_{init} to its final value δ_{final} as the algorithm progresses. This configuration allows HSO to explore the search space more extensively in the initial iterations while gradually focusing on exploiting the current solutions as it approaches the maximum number of iterations.

If mutation occurs (with a probability equal to $\lambda_{\text{mut}}^{\text{iter}}$), a random perturbation $\Delta \mathbf{x}_i$ is added to the agent's position:

$$\Delta \mathbf{x}_i = \delta_{\text{mut}}^{\text{iter}} \cdot \mathcal{N}(0, 1) \quad (11)$$

where $\mathcal{N}(0, 1)$ is a standard normal distribution, and $\Delta \mathbf{x}_i$ is the mutation vector that modifies the current position of the agent $\mathbf{x}_i$.

The updated position of agent i is then calculated as:

$$\mathbf{x}'_i = \mathbf{x}_i + \Delta \mathbf{x}_i \quad (12)$$

where $\mathbf{x}'_i$ is the new position of the agent after mutation.

By following these steps, the HSO algorithm effectively balances global and local search strategies, ensuring robust optimization performance across various problem domains. The pseudocode and flowchart of the proposed algorithm is presented in [Algorithm 1](#), and [Fig. 1](#), respectively.

In summary, the HSO algorithm's holistic approach, combined with adaptive mechanisms, provides a robust framework for solving optimization problems effectively, ensuring thorough exploration and efficient exploitation of the search space.

Algorithm 1 Holistic Swarm Optimization (HSO).

- 1: **Initialization:**
 - 2: Initialize the number of search agents (n), maximum iterations (iter_{max}), number of optimization variables (m), and other necessary parameters.
 - 3: Randomly generate initial positions $\mathbf{x}_i$ for all search agents within the search space.
 - 4: **for** $\text{iter} = 1$ to iter_{max} **do**
 - 5: Evaluate the fitness $f(\mathbf{x}_i)$ of each search agent.
 - 6: Compute the RMS of the fitness values using Equation (1).
 - 7: Use simulated annealing-based selection to accept or reject new positions based on fitness using Equations (5)–(8).
 - 8: Update the best solution found so far.
 - 9: Record the best cost for performance tracking.
 - 10: Calculate the displacement coefficients c_i using Equations (2) and (3).
 - 11: Update the positions of the search agents using Equation (4).
 - 12: Apply adaptive mutation to $\mathbf{x}_i$ using Equations (9)–(12).
 - 13: **end for**
 - 14: **End:**
 - 15: Output the best solution and its corresponding fitness value.
-

3. Computational performance of HSO on benchmark functions

3.1. HSO performance on CEC 2005 benchmarks

To evaluate the proposed HSO algorithm's performance, we analyzed 14 real-world, shifted functions of 30 dimensions (comprising Unimodal, Multimodal, and Expanded Multimodal categories), as outlined in CEC 2005, with data sourced from Suganthan et al. [20]. For each algorithm, the total number of function evaluations (NFE) was limited to 300,000, with the HSO algorithm operating with a population size of 30. Convergence curves were generated over 10,000 iterations. The simulated annealing (SA) selection operator was configured with an initial temperature of 3000 and a cooling rate of 0.995. For the adaptive mutation operator, the initial and

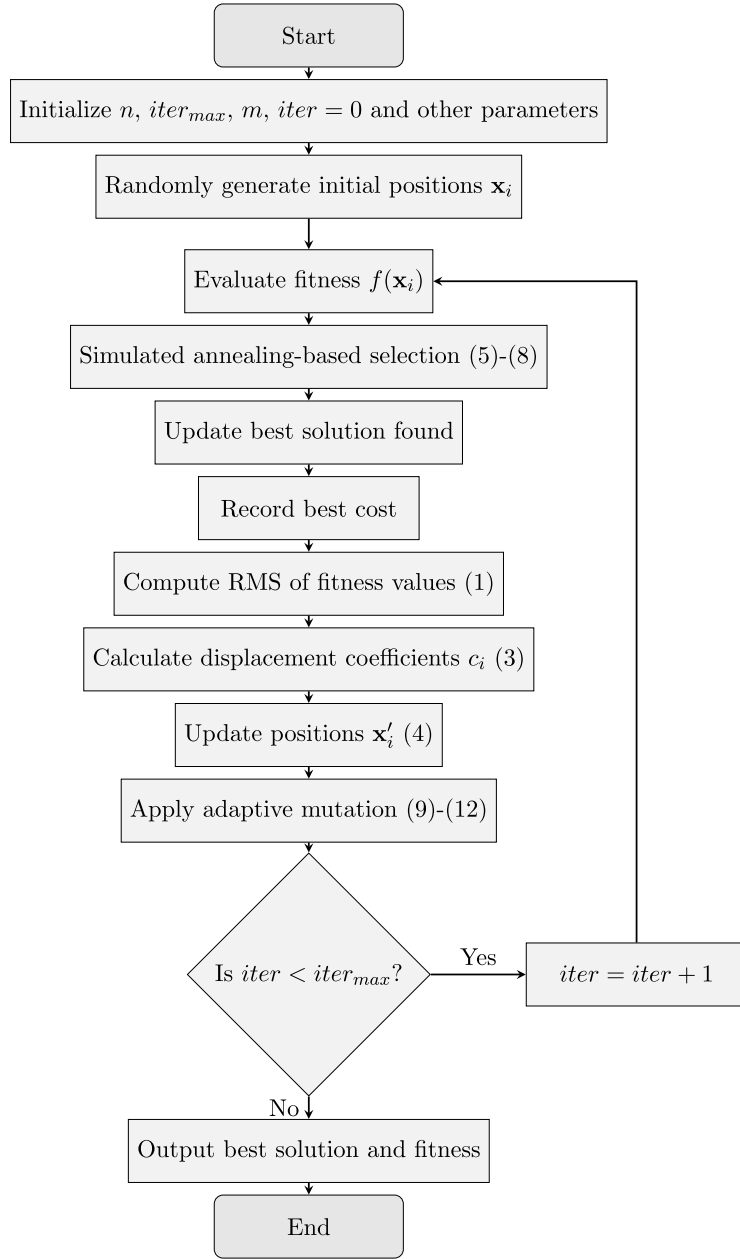


Fig. 1. Flowchart of the Holistic Swarm Optimization (HSO) Algorithm.

final mutation rates were set to 0.5 and 0.1, respectively, with mutation step sizes starting at 0.3 and reducing to 0.1. The impact of these parameters on HSO's performance is thoroughly analyzed in Section 3.2.1. Each algorithm was independently run 30 times per function to obtain optimal solution and statistical results.

Table 1 presents a summary of the results, including the mean value, standard deviation, and ranking of each algorithm. In this table, ' N_b ', ' N_w ', and ' N_e ' denote the instances where the other algorithm achieved a better, a worse, and an equal result with respect to HSO, respectively.

The results presented in Table 1 showcase the comparative performance of the Holistic Swarm Optimization (HSO) algorithm against several other prominent algorithms— Gray Wolf Optimizer (GWO) [21], Greedy Non-Hierarchical GWO (GNHGWO) [22], Rao-1 [16], Rao-2 [16], Rao-3 [16], and FISA [17]—on the 30-D real-parameter test functions of CEC 2005. HSO consistently demonstrates competitive results, often outperforming other algorithms across multiple functions. Specifically, HSO secured the top rank for functions F3, F4, F5, F6, and F12 with a small standard deviation, indicating its robustness and effectiveness in handling complex optimization problems, particularly those involving nonlinear and nonconvex landscapes. Although HSO exhibited slightly lower

Table 1

Comparative performance of HSO and other algorithms on the 30-D real-parameter test functions of CEC 2005.

Function	GWO [22] Mean Std Dev Rank	GNHGWO [22] Mean Std Dev Rank	Rao-1 [17] Mean Std Dev Rank	Rao-2 [17] Mean Std Dev Rank	Rao-3 [17] Mean Std Dev Rank	FISA [17] Mean Std Dev Rank	HSO Mean Std Dev Rank
F1	2.03e+03 2.67e+03 6	6.45e-01 6.71e-01 5	2.88e-29 7.64e-29 1	2.39e-05 3.43e-05 3	4.11e+03 5.53e+02 7	3.37e-27 5.54e-27 2	1.34e-03 3.56e-03 4
F2	1.34e+04 5.95e+03 6	7.50e+02 5.44e+02 4	3.32e-07 8.76e-07 3	7.53e+03 2.52e+03 5	1.51e+04 2.91e+03 7	2.04e-26 3.89e-27 1	1.08e-17 1.59e-17 2
F3	1.78e+07 7.86e+06 5	4.59e+06 1.76e+06 2	1.61e+07 8.89e+06 4	7.02e+07 2.19e+07 6	8.41e+07 2.36e+07 7	5.83e+06 3.84e+06 3	1.52e+06 9.53e+05 1
F4	1.41e+04 2.14e+03 5	1.11e+03 7.33e+02 4	2.18e+02 5.09e+02 3	2.09e+04 8.84e+03 6	2.11e+04 8.43e+03 7	4.01e-05 9.75e-05 2	9.41e-09 1.87e-08 1
F5	5.44e+03 3.65e+03 6	8.25e+02 1.29e+02 2	3.63e+03 1.70e+03 5	3.35e+03 2.44e+03 4	5.56e+03 2.07e+03 7	2.34e+03 9.42e+03 3	2.44e+02 6.60e+02 1
F6	1.06e+08 2.26e+08 6	8.74e+03 1.79e+04 4	2.19e+01 3.54e+01 3	5.55e+06 1.47e+07 5	1.09e+08 2.51e+07 7	4.01e+00 4.96e+07 2	8.00e-01 1.66e+00 1
F7	9.84e+01 8.44e+01 6	1.92e+00 4.75e-01 5	1.98e-02 1.36e-02 1	2.64e-01 2.50e-01 3	3.03e+02 5.19e+01 7	2.88e-02 1.32e-02 2	6.61e-01 1.44e-01 4
F8	2.10e+01 4.06e-02 2	2.09e+01 3.82e-02 1	2.09e+01 5.39e-02 1	2.09e+01 3.42e-05 1	2.09e+01 5.69e-02 1	2.09e+01 5.25e-02 1	2.09e+01 1 1
F9	1.05e+02 1.48e+01 3	1.32e+01 3.09e+00 1	1.63e+02 5.71e+01 4	2.02e+02 1.44e+01 6	2.33e+02 1.21e+01 7	1.91e+02 2.00e+01 5	7.95e+01 4.74e+01 2
F10	2.07e+02 6.74e+01 4	6.99e+01 7.10e+01 1	2.28e+02 2.15e+01 5	2.34e+02 1.48e+01 6	3.03e+02 2.60e+01 7	1.78e+02 1.99e+01 3	1.53e+02 1.22e+01 2
F11	1.87e+01 2.53e+00 1	2.82e+01 1.19e+01 2	4.02e+01 1.25e+00 7	3.93e+01 5.42e-01 6	3.89e+01 1.11e+00 5	3.77e+01 1.65e+00 4	3.76e+01 1.83e+00 3
F12	9.29e+04 3.54e+04 5	1.68e+04 1.63e+04 2	6.11e+04 5.85e+04 4	1.93e+05 1.60e+05 7	1.62e+05 6.39e+04 6	5.28e+04 5.05e+04 3	1.51e+04 2.52e+04 1
F13	3.83e+00 7.07e-01 2	2.93e+00 4.56e-01 1	1.70e+01 8.39e-01 5	1.78e+01 1.61e+00 6	3.38e+01 3.59e+00 7	1.50e+01 1.23e+00 4	1.44e+01 1.48e+00 3
F14	1.16e+01 4.28e-01 2	1.09e+01 3.03e-01 1	1.33e+01 2.29e-01 5	1.34e+01 2.08e-01 6	1.34e+01 1.16e-01 6	1.27e+01 2.02e-01 3	1.29e+01 3.41e-01 4
$N_b/N_w/N_e$	3/11/0	5/8/1	2/11/1	2/11/1	0/13/1	4/9/1	–

performance on remaining functions, the differences were minimal, and HSO surpassed all other algorithms in overall mean rank, which will be further evaluated using statistical tests.

In terms of mean performance, HSO achieved the lowest mean fitness values for many functions, suggesting superior convergence properties. For instance, in function F4, HSO achieved a mean fitness value of $9.41e^{-09}$, which is significantly lower than the closest competitor. The standard deviation values for HSO were minimal across most functions, indicating consistent performance and stability across multiple runs. For example, in function F3, HSO's standard deviation was $9.53e+05$ compared to GNHGWO's $1.76e+06$ (second-best one), reflecting the algorithm's superior stability. This consistency is further highlighted in functions like F6, where HSO's standard deviation was 1.66, compared to FISA's $4.96e+07$ (second-best one), demonstrating the algorithm's reliability in producing high-quality solutions with much less variability. HSO maintains this stable behavior even on functions where it does not achieve the top rank, such as F2 and F7.

Furthermore, the Friedman ranking test results presented in Table 2 provide additional evidence of HSO's superior performance. According to the results, HSO achieved the best overall ranking, with a mean rank of 2.50, outperforming all other algorithms considered. GNHGWO and FISA followed with mean ranks of 2.64 and 2.86, respectively, while the Rao algorithms (Rao-1, Rao-2, and Rao-3) and GWO ranked lower, with GWO having a mean rank of 4.57. The Friedman test results statistically affirm that HSO is the most effective algorithm among those tested, highlighting its capability to consistently achieve optimal solution across a broad range of complex optimization problems.

Table 2

Average ranking of different algorithms according to the Friedman test performed on the results for the CEC 2005 benchmark suite.

Algorithm	RankT	Mean rank
HSO	1	2.50
GNHGWO	2	2.64
FISA	3	2.86
Rao-1	4	3.79
GWO	5	4.57
Rao-2	6	5.18
Rao-3	7	6.46

Table 3

The Wilcoxon signed-rank test results between HSO and other algorithms for the CEC 2005 benchmark suite.

i	j	MoNR	MoPR	SoNR	SoPR	F_{ij}^a	F_{ji}^a	p -value	0.95 Confidence Interval
HSO	GWO	8.7273	3.0000	96.0000	9.0000	11	3	4.03e−03	[−8.14e+06, −4.36e+01]
HSO	GNHGWO	9.2500	5.1667	74.0000	31.0000	8	6	1.94e−01	[−4.37e+03, 5.44e+00]
HSO	Rao-1	8.6364	3.3333	95.0000	10.0000	11	3	5.25e−03	[−2.30e+04, −1.30e+00]
HSO	Rao-2	9.0000	2.0000	99.0000	6.0000	11	3	1.71e−03	[−2.78e+06, −4.03e+01]
HSO	Rao-3	8.0000	1.0000	104.0000	1.0000	13	1	2.44e−04	[−4.13e+07, −1.52e+02]
HSO	FISA	9.2222	4.4000	83.0000	22.0000	9	5	5.80e−02	[−1.89e+04, 1.08e−17]

^a F_{ij} represents the number of times HSO outperforms the other algorithm for the CEC 2005 benchmark suite. F_{ji} represents the number of times HSO gives a result worse than the other algorithm.

The results of the Wilcoxon signed-rank test are presented in Table 3. In this table, SoNR and SoPR represent the combined negative and positive ranks, respectively (i.e. sum of the ranks with HSO outperforming the other algorithm or vice versa, respectively). Similarly, MoNR and MoPR represent the average negative and positive ranks, respectively. It is observed that the p -values between HSO and GWO, Rao-1, Rao-2, and Rao-3 are all below the significance level of 0.05, indicating that HSO is significantly different from these algorithms and superior in solving the CEC 2005 benchmark functions, as evidenced by the higher SoNR values compared to SoPR values.

Conversely, when comparing HSO to FISA and GNHGWO, the p -values exceed the 0.05 significance level. This indicates that the Wilcoxon signed-rank test does not demonstrate a statistically significant difference between HSO and these algorithms for the function sets studied. Nonetheless, the SoNR values for both cases remain acceptably higher than the SoPR values, and both cases exhibit F_{ij} values greater than F_{ji} values. This suggests that HSO outperforms FISA and GNHGWO, even though the differences are not statistically significant according to the Wilcoxon signed-rank test at the 0.95 confidence interval.

Furthermore, the 0.95 confidence intervals for these cases, which include zero but possess a negative lower limit well below zero and a positive upper limit slightly above zero, indicate that while there is no statistically significant difference between HSO and these algorithms at the 0.05 level, the data suggests a potential advantage for HSO. The negative lower limit implies that HSO may still perform better than FISA and GNHGWO in certain scenarios, supported by the observed higher SoNR values. Therefore, while the Wilcoxon signed-rank test encourages caution in claiming superiority, the overall pattern of results suggests that HSO is a strong performer among the evaluated algorithms.

Moreover, these findings highlight the importance of considering both statistical and practical significance. The results exhibit consistent trends that may not be statistically robust yet demonstrate a performance edge for HSO. This observation has motivated us to further investigate the performance of the proposed algorithm on the CEC 2014 benchmark suite, which includes 30 shifted high-dimensional functions.

3.2. HSO performance on CEC 2014 benchmarks

To further demonstrate the effectiveness of the HSO algorithm against other existing well-known methods, we evaluated 30 test functions from the CEC 2014 benchmark suite, each with 30 dimensions [17,23], with data sourced from Liang et al. [24]. The optimal value for all functions is 0. Again, The population size was consistently set at 30, with each algorithm running for 10,000 iterations, leading to a total of 300,000 function evaluations (NFE). Each function was independently executed 30 times per algorithm. The simulated annealing (SA) selection operator used an initial temperature of 3000 and a cooling rate of 0.995, while the adaptive mutation operator started with a mutation rate of 0.5, decreasing to 0.1, and mutation step sizes reducing from 0.3 to 0.1. The effects of these parameters on HSO's performance are comprehensively discussed in Section 3.2.1.

Table 4 summarizes the average values, standard deviations, and the mean-based rank of HSO obtained from 30 runs compared to other competing algorithms including Particle Swarm Optimization [25], Fully Informed Search Algorithm (FISA) [17], Rao-1 [16], Pbest guided Differential WOA (PDWOA) [23], JAYA algorithm [15], a Greedy Non-hierarchical Gray Wolf Optimizer (G-NHGWO) [22], Salp Swarm Algorithm (SSA) [26], Hunger Games Search (HGS) algorithm [27], and island-based Enhanced Crow Search

Table 4

The optimal results obtained by HSO and Other algorithms on the 30-D real-parameter test functions of CEC 2014.

Function	PSO [23] Mean Std Dev Rank	PDWOA-rnd [23] Mean Std Dev Rank	Rao-1 [23] Mean Std Dev Rank	FISA [17] Mean Std Dev Rank	JAYA [28] Mean Std Dev Rank	SSA [28] Mean Std Dev Rank	HGS [28] Mean Std Dev Rank	iECOA [28] Mean Std Dev Rank	GNHGW0 [22] Mean Std Dev Rank	HSO Mean Std Dev Rank
F1	4.38e+07 1.81e+07 9	3.15e+06 1.75e+06 3	9.58e+06 3.08e+06 5	2.80e+06 2.15e+06 2	1.14e+08 3.64e+07 10	1.42e+07 9.39e+06 7	1.72e+07 1.10e+07 8	5.00e+06 1.98e+06 4	1.00e+07 4.23e+06 6	8.63e+05 7.19e+05 1
F2	9.83e+08 5.36e+08 9	1.42e+04 1.32e+04 6	1.21e+02 1.13e+02 3	2.67e-01 2.91e-01 1	1.01e+10 2.17e+09 10	9.71e+03 9.91e+03 5	7.37e+08 8.77e+08 8	3.87e+03 3.08e+03 4	1.82e+07 2.39e+07 7	9.08e+00 1.03e+01 2
F3	2.84e+04 1.40e+04 8	3.51e+03 3.56e+03 4	4.49e+03 2.23e+03 6	3.11e+01 3.19e+01 2	1.17e+05 2.17e+04 10	3.97e+04 1.49e+04 9	1.13e+04 4.51e+03 7	3.53e+03 1.06e+03 5	2.45e+03 2.08e+03 3	1.12e+00 1.97e+00 1
F4	2.61e+02 8.36e+01 9	1.23e+02 3.19e+01 5	6.17e+01 5.15e+01 3	2.44e+01 3.12e+01 2	6.90e+02 1.56e+02 10	1.32e+02 4.00e+01 7	1.66e+02 5.44e+01 8	9.40e+01 3.49e+01 4	1.25e+02 9.68e+00 6	9.99e+00 2.39e+01 1
F5	2.04e+01 6.00e-02 3	2.02e+01 1.50e-01 2	2.09e+01 6.76e-02 5	2.09e+01 6.96e-02 6	2.10e+01 5.60e-02 8	2.00e+01 1.15e-01 1	2.00e+01 7.06e-02 1	2.00e+01 5.62e-04 1	2.10e+01 4.54e-02 7	2.09e+01 7.80e-02 4
F6	2.95e+01 2.70e+00 8	2.63e+01 3.33e+00 6	1.53e+01 7.28e+00 4	9.54e+00 2.44e+00 2	3.70e+01 1.53e+00 9	1.90e+01 5.46e+00 5	1.90e+01 2.56e+00 5	2.70e+01 1.81e+00 7	2.11e+00 7.45e-01 1	1.11e+01 2.94e+00 3
F7	1.50e+01 1.36e+01 8	9.00e-02 1.00e-01 4	5.91e-03 5.47e-03 2	6.89e-03 9.19e-03 3	3.20e+01 6.50e+00 9	0.00e+00 9.63e-03 1	5.00e+00 5.47e+00 7	0.00e+00 5.53e-03 1	1.15e+00 3.30e-01 6	8.31e-01 1.95e-01 5
F8	1.46e+02 2.85e+01 6	6.64e+01 1.68e+01 4	1.71e+02 4.58e+01 8	1.84e+02 1.08e+01 9	2.50e+02 1.58e+01 10	1.47e+02 2.81e+01 7	6.20e+01 2.03e+01 3	1.03e+02 1.16e+01 5	1.00e+01 2.18e+00 1	5.90e+01 1.49e+01 2
F9	1.78e+02 2.72e+01 6	2.05e+02 5.24e+01 8	2.24e+02 1.91e+01 9	1.84e+02 8.10e+00 7	2.80e+02 1.56e+01 10	1.60e+02 3.64e+01 5	1.50e+02 2.88e+01 4	1.10e+02 1.35e+01 3	2.36e+01 3.67e+01 1	7.35e+01 2.05e+01 2
F10	4.37e+03 6.39e+02 7	6.27e+02 4.05e+02 1	5.98e+03 4.07e+02 8	6.11e+03 3.69e+02 9	6.32e+03 4.04e+02 10	3.62e+03 8.04e+02 6	1.16e+03 3.61e+02 2	2.88e+03 5.21e+02 5	2.27e+03 1.19e+03 4	2.02e+03 2.28e+02 3
F11	4.90e+03 9.12e+02 7	4.11e+03 8.53e+02 6	6.93e+03 4.75e+02 9	6.57e+03 2.95e+02 8	7.22e+03 3.59e+02 10	3.98e+03 8.04e+02 5	2.78e+03 5.60e+02 2	3.02e+03 3.85e+02 3	3.12e+03 2.20e+03 4	2.37e+03 4.39e+02 1
F12	1.36e+00 3.90e-01 4	1.15e+00 6.70e-01 3	2.35e+00 3.79e-01 5	2.40e+00 2.38e-01 6	0.00e+00 2.95e-01 1	0.00e+00 1.51e-01 1	0.00e+00 1.13e-01 1	0.00e+00 1.84e-01 1	2.52e+00 3.68e-01 7	2.92e-02 1.35e-02 2
F13	5.40e-01 7.00e-02 7	5.10e-01 1.30e-01 5	5.11e-01 1.24e-01 6	3.42e-01 5.83e-02 3	0.00e+00 3.60e-01 1	0.00e+00 1.18e-01 1	0.00e+00 1.41e-01 1	0.00e+00 7.29e-02 1	1.71e-01 3.51e-02 2	3.54e-01 1.19e-01 4
F14	1.63e+00 2.84e+00 7	3.00e-01 5.00e-02 3	5.39e-01 2.94e-01 5	2.83e-01 3.43e-02 2	2.00e+01 3.84e+00 8	0.00e+00 1.91e-01 1	0.00e+00 6.76e-01 1	0.00e+00 4.43e-02 1	5.61e-01 8.13e-02 6	3.94e-01 1.16e-01 4
F15	1.05e+02 6.57e+01 8	4.55e+01 1.21e+01 7	1.70e+01 2.40e+00 5	1.54e+01 1.47e+00 4	1.30e+02 5.15e+01 9	1.00e+01 4.93e+00 3	3.00e+01 3.65e+01 6	1.00e+01 3.00e+00 3	9.63e+00 4.32e+00 2	3.18e+00 1.17e+00 1
F16	1.28e+01 5.60e-01 6	1.17e+01 4.00e-01 3	1.28e+01 2.66e-01 7	1.19e+01 2.73e-01 4	1.00e+01 1.55e-01 2	1.00e+01 5.06e-01 2	1.00e+01 3.34e-01 2	1.00e+01 4.23e-01 2	9.62e+00 5.20e-01 1	1.27e+01 4.56e-01 5
F17	1.14e+06 8.15e+05 8	7.82e+05 3.34e+05 6	9.65e+05 4.37e+05 7	3.90e+05 3.26e+05 4	7.34e+06 2.98e+06 10	6.30e+05 3.69e+05 5	2.18e+06 1.47e+06 9	6.48e+04 5.52e+04 2	2.82e+05 1.61e+05 3	5.67e+04 4.46e+04 1
F18	5.40e+03 2.45e+03 4	4.34e+03 4.79e+03 3	1.99e+05 4.49e+05 9	1.13e+04 9.52e+03 6	5.11e+07 6.47e+07 10	1.17e+04 8.88e+03 7	1.02e+04 1.12e+04 5	2.40e+02 2.93e+01 1	1.29e+04 2.78e+04 8	1.52e+03 1.47e+03 2
F19	3.23e+01 2.50e+01 7	2.39e+01 2.67e+01 5	1.06e+01 1.68e+00 3	2.86e+01 3.23e+01 6	5.00e+01 3.16e+01 8	2.00e+01 1.48e+01 4	2.00e+01 3.11e+01 4	2.00e+01 2.08e+00 4	9.72e+00 1.91e+00 2	9.57e+00 1.44e+00 1
F20	1.04e+04 5.33e+03 7	7.76e+03 4.60e+03 6	1.46e+03 8.08e+02 4	3.49e+02 1.19e+02 2	3.50e+04 1.65e+04 10	1.21e+04 6.27e+03 9	1.11e+04 4.65e+03 8	3.60e+02 1.20e+02 3	2.91e+03 2.35e+03 5	3.32e+02 9.45e+01 1
F21	1.13e+05 1.27e+05 3	4.87e+05 2.88e+05 8	2.88e+05 1.87e+05 7	1.53e+05 1.22e+05 4	1.73e+06 8.20e+05 10	2.16e+05 1.34e+05 6	5.05e+05 4.70e+05 9	2.26e+04 1.73e+04 2	1.79e+05 1.22e+05 5	2.15e+04 2.09e+04 1
F22	7.26e+02 1.21e+02 7	9.87e+02 1.88e+02 10	5.87e+02 1.67e+02 5	3.79e+02 1.33e+02 2	8.00e+02 1.50e+02 8	5.70e+02 1.80e+02 4	6.20e+02 1.97e+02 6	4.40e+02 1.51e+02 3	1.16e+02 5.37e+01 1	9.01e+02 2.15e+02 9
F23	3.40e+02 7.70e+00 8	3.15e+02 2.00e-02 2	3.15e+02 8.78e-13 3	3.15e+02 5.95e-13 3	3.70e+02 1.41e+01 9	3.30e+02 5.48e+00 7	2.00e+02 0.00e+00 1	3.20e+02 5.92e-01 6	3.17e+02 2.49e-01 5	3.16e+02 8.12e-01 4
F24	2.55e+02 7.98e+00	2.18e+02 1.12e+01	2.40e+02 9.10e+00	2.29e+02 7.22e+00	2.70e+02 6.15e+00	2.30e+02 5.60e+00	2.00e+02 1.02e-07	2.10e+02 5.78e+00	2.00e+02 4.84e-03	2.32e+02 1.45e+01

Table 4
Continue

Function	PSO [23] Mean Std Dev Rank	PDWOA-rnd [23] Mean Std Dev Rank	Rao-1 [23] Mean Std Dev Rank	FISA [17] Mean Std Dev Rank	JAYA [28] Mean Std Dev Rank	SSA [28] Mean Std Dev Rank	HGS [28] Mean Std Dev Rank	iECOA [28] Mean Std Dev Rank	GNHGW0 [22] Mean Std Dev Rank	HSO Mean Std Dev Rank
F25	9 2.22e+02 4.31e+00 7	4 2.23e+02 1.63e+01 8	8 2.06e+02 3.14e+00 5	5 2.04e+02 9.16e-01 2	10 2.30e+02 5.05e+00 9	6 2.10e+02 4.06e+00 6	1 2.00e+02 0.00e+00 1	3 2.00e+02 3.19e+00 1	2 2.05e+02 7.72e-01 4	7 2.05e+02 1.27e+00 3
F26	1.73e+02 4.93e+01 8	1.15e+02 3.76e+01 5	1.24e+02 7.57e+01 6	1.10e+02 3.15e+01 3	1.00e+02 2.04e-01 1	1.00e+02 1.12e-01 1	1.40e+02 4.87e+01 7	1.00e+02 7.01e-02 1	1.07e+02 2.58e+01 2	1.14e+02 3.50e+01 4
F27	9.04e+02 3.26e+02 8	1.11e+03 8.11e+01 9	7.64e+02 1.90e+02 7	6.23e+02 9.91e+01 5	1.11e+03 1.92e+02 9	7.40e+02 1.95e+02 6	2.00e+02 0.00e+00 1	4.10e+02 4.41e+00 3	3.54e+02 2.44e+01 2	5.79e+02 1.25e+02 4
F28	3.45e+03 1.67e+03 10	1.88e+03 8.69e+02 8	1.24e+03 2.52e+02 6	1.07e+03 1.45e+02 3	1.37e+03 2.58e+02 7	1.21e+03 3.46e+02 5	2.00e+02 0.00e+00 1	2.99e+03 6.28e+02 9	8.63e+02 3.55e+01 2	1.09e+03 4.07e+02 4
F29	2.09e+07 3.46e+07 10	7.54e+06 5.27e+06 8	9.51e+06 7.77e+06 9	1.97e+06 4.16e+06 5	3.31e+06 4.13e+06 7	2.63e+06 4.74e+06 6	2.17e+03 2.52e+03 2	3.88e+03 1.92e+03 3	2.29e+04 2.22e+03 4	2.07e+03 9.95e+02 1
F30	5.10e+04 7.03e+04 10	3.91e+03 1.91e+03 2	2.21e+04 2.27e+04 7	4.81e+03 2.40e+03 3	2.31e+04 1.39e+04 8	2.34e+04 1.24e+04 9	6.54e+03 3.71e+03 4	1.23e+04 4.56e+03 6	1.09e+04 4.13e+03 5	1.91e+03 9.35e+02 1
$N_b/N_w/N_e$	2/28/0	7/23/0	3/27/0	12/18/0	5/25/0	9/21/0	12/18/0	12/18/0	10/20/0	-

Algorithm (iECOA) [28]. The last row of the table shows the number of functions where each algorithm attained a better (' N_b '), a worse (' N_w '), and an equal (' N_e ') result with respect to HSO.

As observed, the results presented in Table 4 demonstrate the performance of the Holistic Swarm Optimization (HSO) algorithm in comparison to other state-of-the-art algorithms across the CEC 2014 benchmark suite. HSO consistently performs well, often achieving the best or near-best results across various functions. For instance, in function F1, HSO outperforms all other algorithms with a mean value of $8.63e+05$ and the lowest standard deviation of $7.19e+05$. Moreover, HSO shows excellent performance in functions such as F3, F4, F11, F15, F17, F19, F20, F21, F29, and F30, where it secures a top rank, demonstrating its effectiveness in handling diverse complex optimization problems.

The standard deviation values further highlight the stability of HSO. In analyzing the results presented in Table 4, it is evident that the HSO algorithm demonstrates a consistently lower standard deviation compared to other algorithms for the majority of the test functions specifically where it ranked first. This lower standard deviation indicates that HSO not only achieves better average performance but also exhibits more stable and reliable results across multiple runs.

For instance, in Function 1 (F1), HSO achieves the best mean performance with a standard deviation of $7.19e+05$, significantly lower than many of the other algorithms, such as iECOA, GNHGW0, and FISA, which have much higher standard deviations of $1.98e+06$, $4.23e+06$, and $3.64e+07$, respectively. This suggests that HSO's solutions are less variable and more consistently near the optimum, contributing to its superior ranking.

Similarly, for Function 2 (F2), HSO again demonstrates a markedly lower standard deviation of $1.03e+01$ compared to the other algorithms, reinforcing its robustness and consistency. This pattern is consistent across other functions, even where HSO is not ranked first, such as F5, F10, F12, and F16, further validating its effectiveness and reliability in consistently finding high-quality solutions with minimal variation. Overall, the results for all functions reflect the algorithm's ability to maintain stability while achieving low objective function values, which is critical for complex optimization tasks.

Additionally, HSO's overall ranking across the functions is indicative of its superior performance. This is further substantiated by the Friedman ranking test results presented in Table 5, where HSO attains the best average rank of 3.40 among all compared algorithms. The close ranking of iECOA with a mean rank of 3.63 indicates stiff competition, yet HSO maintains a leading edge.

Table 5
Average ranking of different algorithms according to the CEC 2014 benchmark functions.

Algorithm	RankT	Mean rank
HSO	1	3.40
iECOA	2	3.63
GNHGW0	3	4.30
HGS	4	4.58
FISA	5	4.78
SSA	6	5.30
PDWOA-rnd	7	5.85
Rao-1	8	6.48
PSO	9	8.00
JAYA	10	8.67

Table 6

The Wilcoxon signed-rank test results between HSO and other algorithms on CEC 2014 functions.

i	j	MoNR	MoPR	SoNR	SoPR	F_{ij}^a	F_{ji}^a	p-value	0.95 Confidence Interval
HSO	PSO	15.9286	9.5000	446.0000	19.0000	28.0000	2.0000	5.72e-07	[-45700, -104]
HSO	PDWOA-rnd	17.9130	7.5714	412.0000	53.0000	23.0000	7.0000	7.91e-05	[-7100, -50.5]
HSO	Rao-1	16.0741	10.3333	434.0000	31.0000	27.0000	3.0000	4.42e-06	[-98800, -56.4]
HSO	FISA	19.9444	8.8333	359.0000	106.0000	18.0000	12.0000	8.14e-03	[-4880, -6.8]
HSO	JAYA	17.5600	5.2000	439.0000	26.0000	25.0000	5.0000	1.99e-06	[-1650000, -149]
HSO	SSA	19.2857	6.6667	405.0000	60.0000	21.0000	9.0000	1.70e-04	[-19800, -53.8]
HSO	HGS	18.5000	11.0000	333.0000	132.0000	18.0000	12.0000	3.84e-02	[-5650, -1.5]
HSO	iECSA	19.5556	9.4167	352.0000	113.0000	18.0000	12.0000	1.28e-02	[-1750, -5.61]
HSO	GNHGWO	16.5500	13.4000	331.0000	134.0000	20.0000	10.0000	4.27e-02	[-10400, -0.121]

^a F_{ij} represents the number of times HSO outperforms the other algorithm. F_{ji} represents the number of times HSO gives a result worse than the other algorithm.

The “ $N_b/N_w/N_e$ ” row in Table 4 also shows that HSO has a favorable balance of wins to losses in head-to-head comparisons. This consistent performance across the CEC 2014 suite, corroborated by statistical ranking, demonstrates HSO’s potential as a robust and versatile optimization tool for a wide range of applications.

Fig. 2 illustrates the convergence behavior of the HSO algorithm compared to GNHGWO, PDWOA, and FISA on selected CEC 2014 benchmark functions. Across all six functions (F2, F10, F17, F20, F23, and F28), HSO consistently shows a faster convergence rate, reaching lower objective function values in fewer iterations than the other algorithms. Notably, HSO’s performance is particularly impressive in the earlier iterations, where it demonstrates a significant advantage in finding near-optimal solution quickly. As the iterations progress, HSO continues to maintain or slightly improve its solution quality, whereas the other algorithms, particularly GNHGWO and FISA, exhibit slower convergence and often stagnate at higher objective function values. This indicates the superior exploitation capabilities of HSO, allowing it to refine solutions effectively during the later stages of the optimization process. Overall, HSO’s balance between exploration and exploitation enables it to outperform the comparative algorithms across different types of test functions, as shown in Fig. 2.

The Wilcoxon signed-rank test results presented in Table 6 provide a statistical analysis of the comparative performance between the Holistic Swarm Optimization (HSO) algorithm and several other optimization algorithms on the CEC 2014 functions. The results indicate that HSO consistently outperforms its competitors, achieving statistically significant margins across all comparisons. For instance, when compared to PSO, HSO attained a p -value of 5.72×10^{-7} with a confidence interval of [-45700, -104], along with SoNR and Fij values of 446 and 28, respectively. These figures are significantly higher than the SoPR and Fji values of 19 and 2, respectively, underscoring HSO’s strong and statistically significant advantage. Similar trends are evident in comparisons with all other algorithms, further highlighting HSO’s robust performance across various optimization tasks in the CEC 2014 suite.

Notably, the Wilcoxon signed-rank test results comparing HSO with FISA and GNHGWO yield p -values of 8.14×10^{-3} and 4.27×10^{-2} , respectively. This indicates that, while HSO did not demonstrate statistically significant differences from these algorithms in the CEC 2005 test functions, it shows significant performance differences in solving the CEC 2014 test functions. Furthermore, the confidence intervals for all algorithms do not include zero, which corroborates the statistical significance of the differences indicated by the p -values. This improvement may be attributed to the doubling of sample sizes for the Wilcoxon test in the CEC 2014 benchmark suite.

While our main comparative analysis focused on solution quality and statistical robustness, we acknowledge that computational time is also an important consideration. In our experiments, HSO required more computational effort per iteration compared to simpler algorithms such as Rao-1 or PSO, due to its full-population RMS coefficient calculation and adaptive control mechanisms. However, the overhead remains modest. This increase is largely offset by the superior convergence rate and fewer required function evaluations for achieving a given target accuracy. Importantly, HSO remains computationally tractable for large-scale continuous optimization and can benefit from parallelization of population-level operations. Future work may include a more detailed profiling of per-function CPU times and evaluation of scalability across different hardware platforms.

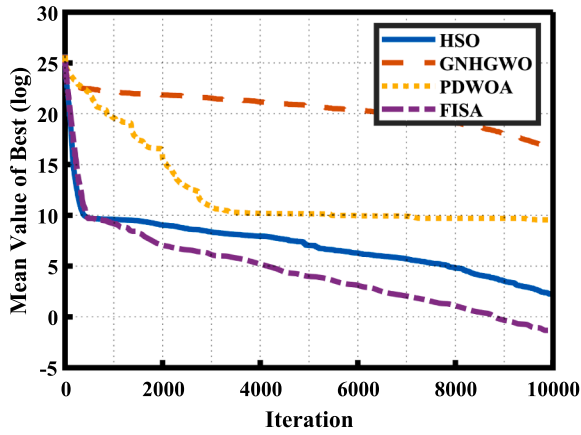
3.2.1. Discussion on exploration-exploitation capability of HSO using population diversity

The population diversity, as measured by the Standard Deviation (St.D.) of Euclidean distance of (d -dimensional) solutions (n search agents) with respect to their overall mean [28], provides insights into the exploration-exploitation capabilities of the optimization algorithms under different settings. It is noteworthy that the Population Diversity Index used in this study is defined as follows:

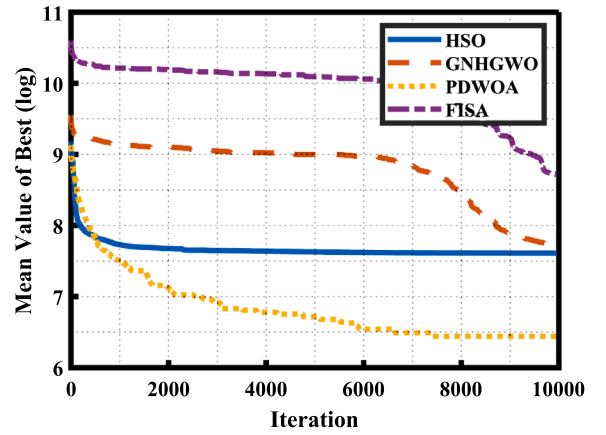
$$PDI = \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^d (x_{ij} - \bar{x}_j)^2} \quad (13)$$

where x_{ij} represents the j th dimension of the i th solution, $\bar{x}_j$ is the mean value of the j th dimension across all solutions, d is the number of dimensions, and n is the total number of search agents.

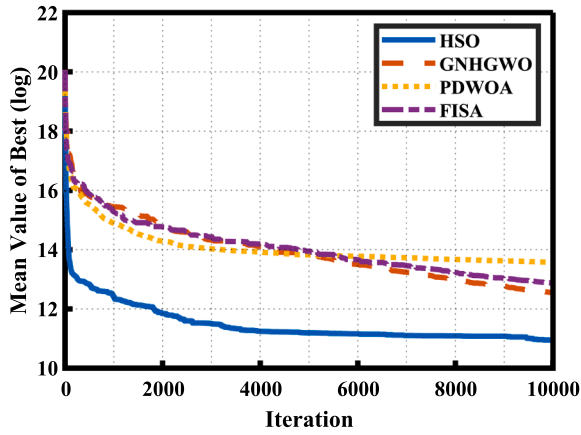
In the context of the figures presented (Fig. 3), the diversity of the population was observed across different mutation cases and Simulated Annealing (SA) cases within two different setups. The detailed configuration of these cases is summarized in Table 7.



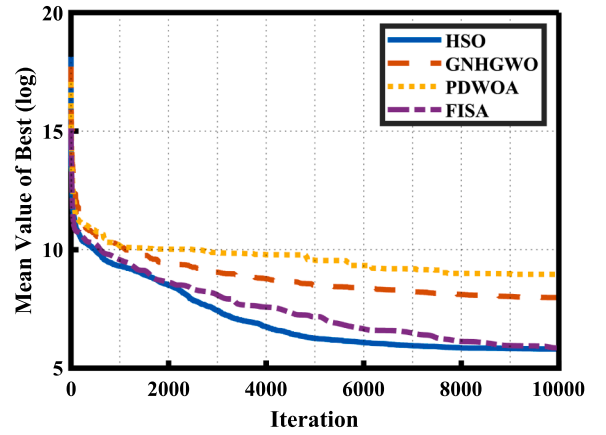
(a) F2 function



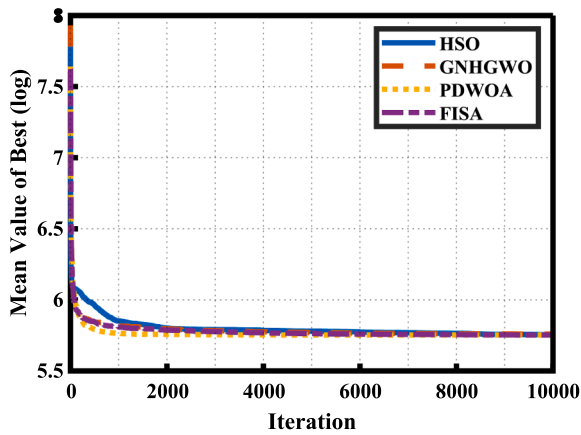
(b) F10 function



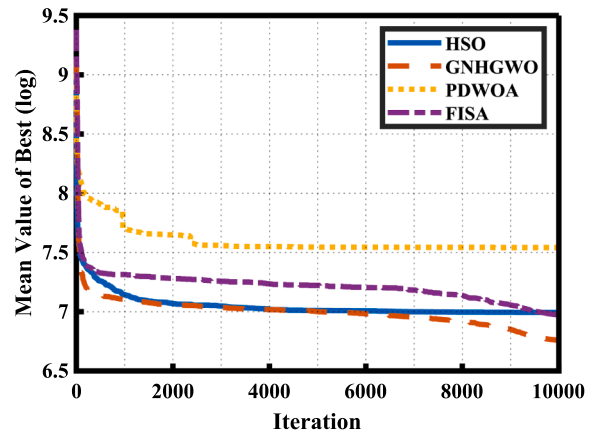
(c) F17 function



(d) F20 function



(e) F23 function



(f) F28 function

Fig. 2. Convergence behavior of HSO compared with GNHGWO, PDWOA, and FISA on selected CEC 2014 benchmark functions: (a) F2, (b) F10, (c) F17, (d) F20, (e) F23, (f) F28.

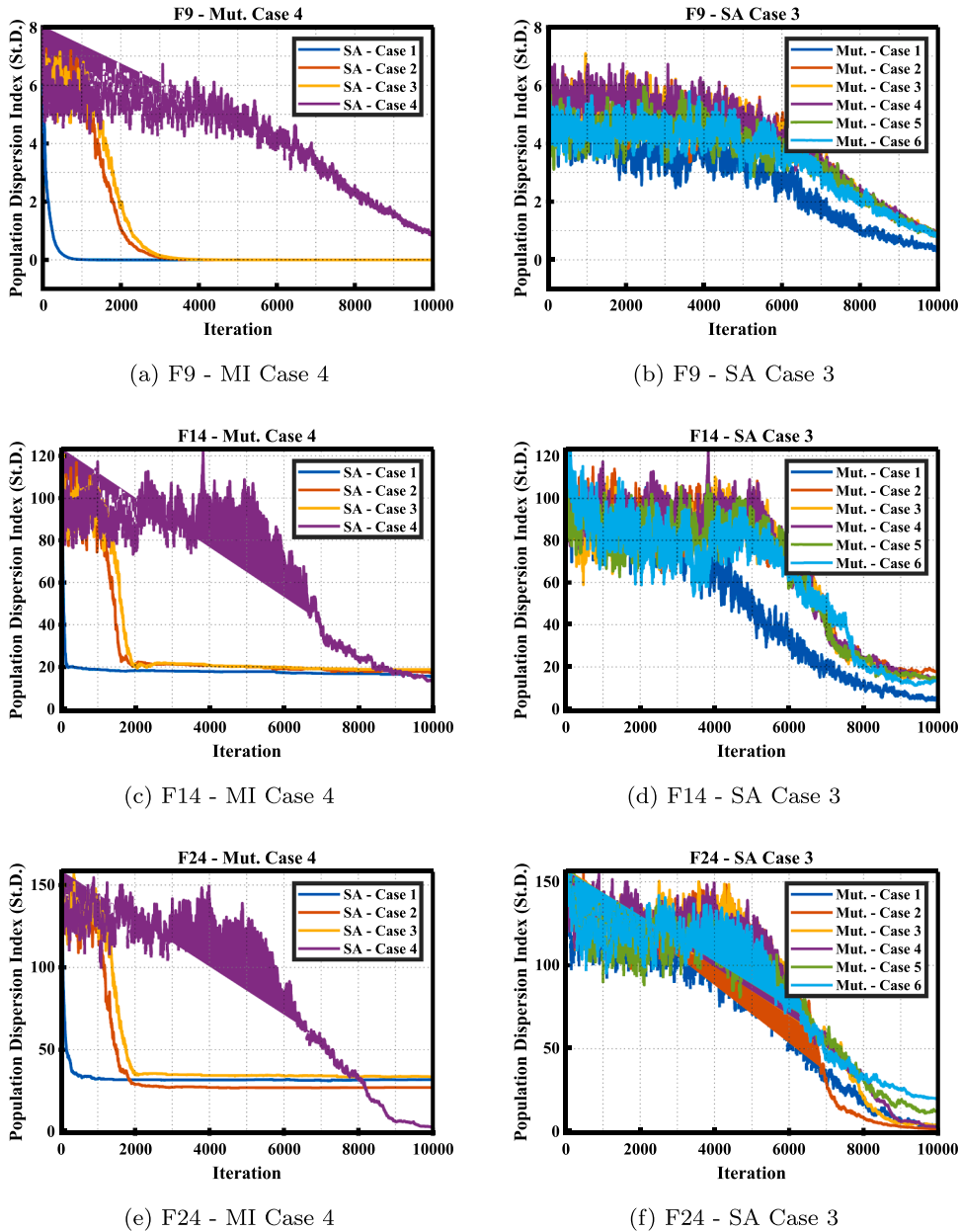


Fig. 3. Convergence behavior for different mutation cases in MI and SA scenarios on selected benchmark functions: (a) F9 - MI Case 4, (b) F9 - SA Case 3, (c) F14 - MI Case 4, (d) F14 - SA Case 3, (e) F24 - MI Case 4, (f) F24 - SA Case 3.

In the case of **SA Case 1**, where both the initial temperature and cooling rate are set to zero, the population diversity remains relatively low, indicating minimal exploration in the search space. This behavior suggests a high likelihood of premature convergence, as the population quickly settles into a limited area of the solution space without significant exploration, resulting in a narrow range of solutions.

When considering **SA Case 2** and **SA Case 3**, where the initial temperature is set to 3000 and 10,000 respectively, and the cooling rate is 0.995, the population diversity is higher in the initial stages, reflecting an increased exploration capacity due to the higher initial temperature. This diversity diminishes as the temperature decreases, indicating that the algorithm concentrates more on exploitation rather than exploration. Finally, **SA Case 4**, with an initial temperature of 3000 and a cooling rate of 0.999, shows a more gradual reduction in population diversity compared to SA Case 3.

On the other hand, **Mutation Case 1**, which involves no mutation, exhibits the lowest population diversity, consistent with the expectation that the lack of mutation leads to rapid convergence and limited exploration.

Table 7
Detailed configuration of Simulated Annealing (SA) and Mutation Cases.

Case	Parameter	Initial Value	Final Value	Description
SA Case 1	T_{initial}	0	N/A	Initial temperature is zero.
	CR	0	N/A	No cooling, temperature remains zero.
SA Case 2	T_{initial}	3000	N/A	Moderate initial temperature.
	CR	0.995	N/A	Fast cooling rate.
SA Case 3	T_{initial}	10,000	N/A	High initial temperature.
	CR	0.995	N/A	Fast cooling rate.
SA Case 4	T_{initial}	3000	N/A	Moderate initial temperature.
	CR	0.999	N/A	Slow cooling rate.
Mutation Case 1	λ_{mut}	N/A	N/A	No mutation.
Mutation Case 2	λ_{mut}	0.1	0.1	Fixed mutation rate.
	δ_{mut}	0.1	0.1	Fixed mutation step size.
Mutation Case 3	λ_{mut}	0.2	0.1	Moderate decreasing mutation rate.
	δ_{mut}	0.2	0.1	Moderate decreasing mutation step size.
Mutation Case 4	λ_{mut}	0.5	0.1	High initial mutation rate.
	δ_{mut}	0.3	0.1	Larger mutation step size.
Mutation Case 5	λ_{mut}	0.5	0.5	Fixed high mutation rate.
	δ_{mut}	0.01	0.001	Small decreasing mutation step sizes.
Mutation Case 6	λ_{mut}	0.2	0.1	Moderate decreasing mutation rate.
	δ_{mut}	0.01	0.001	Small decreasing mutation step sizes.

Mutation Case 2 and **Mutation Case 3** demonstrate a moderate level of population diversity, with the fixed-parameter mutation (Case 2) showing slightly less diversity compared to the varying-parameter mutation (Case 3). The adjustments in mutation rates and steps in Mutation Case 3 allow for a more varied search, reflected in higher diversity.

In **Mutation Case 4**, the higher initial mutation rate (0.5) and step size (0.3) lead to significantly higher population diversity, which gradually decreases as the final mutation rate and step size are reduced. This behavior highlights the impact of aggressive mutation strategies on maintaining population diversity during the search process.

Lastly, **Mutation Case 5** and **Mutation Case 6** demonstrate differing effects based on mutation parameters. Mutation Case 5, with a high and consistent mutation rate (0.5) and very small step sizes, leads to moderate diversity, while Mutation Case 6, with slightly reduced mutation rates and steps, results in less diversity, particularly towards the end of the optimization process.

4. Numerical results of HSO in addressing engineering optimization problems

To evaluate the performance of the HSO algorithm, six well-known engineering design problems have been selected: the optimal design of the pressure vessel design, tension/compression spring design, welded beam design, cantilever beam design, and three-bar truss design optimization. These problems are widely recognized benchmarks in the field of engineering optimization. The population size for HSO was set to 50, and the number of iterations for each run was fixed at 2000. Additionally, each optimization task was executed in 30 independent Monte Carlo runs. Since the number of iterations for the engineering problem is set to a much lower value compared to that of benchmark functions, we reduced the cooling rate of the SA operator of the proposed HSO from 0.995 to 0.99 so as to improve the algorithm's capability to exploit with lowering the rate of the exploration imposed by SA operator. The parameters of the mutation operator were set to the same values as those chosen for the CEC benchmark functions.

4.1. Optimal design of a pressure vessel

The primary goal of this problem is to identify the optimal values for two discrete (x_1 and x_2) and two continuous (x_3 and x_4) decision variables that will minimize the cost of a pressure vessel (refer to Fig. 4). The optimization is subject to three linear and one nonlinear inequality constraints. The variables being optimized include the thickness of the shell (x_1 or T_s), the thickness of the head

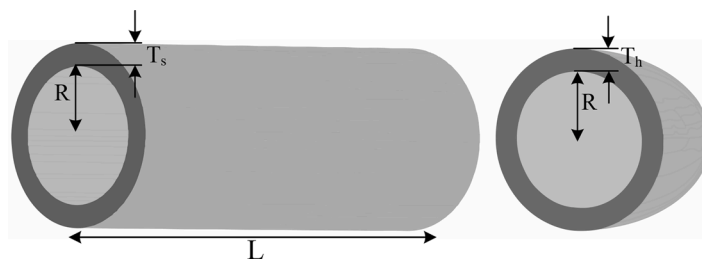


Fig. 4. Schematic illustration of design parameters of pressure vessel.

Table 8

Best statistical results of various algorithms for pressure vessel design problem [45].

Methods	Best	Mean	Worst	Std.
Proposed HSO	6059.71334	6207.85426	6411.11725	158.213319
AO	6060.226707	6572.408065	7547.794432	474.7811046
AOA	7177.685796	10134.5141	21033.80706	3059.849954
CA	6058.719878	6249.526308	7273.278824	303.558052
COA	6340.732653	7846.678347	8548.44731	447.1938273
COOT	6058.720225	6406.088039	7352.612207	315.2009216
EO	6058.71988	6518.793086	7544.492518	512.0568854
GWO	6058.801906	6219.805205	7365.851023	356.5801053
HHO	6091.615788	6792.936176	7544.492518	365.7904697
MFO	6058.72346	6722.668915	7544.492518	508.6435023
MVO	6091.584113	6741.27325	7419.29367	421.1743251
POA	6058.719878	6224.114264	7273.278824	284.930973
SAO	8794.883762	13888.48817	21019.53969	3282.14935
SCA	6192.682512	6929.151724	8752.385245	588.2808671
SMA	6058.719892	6596.192699	7544.492519	549.8023233
SOA	6059.667071	6358.863021	7546.780769	466.1025508
TSO	6879.866741	8704.301853	13069.25436	1582.300611
WOA	6059.823537	6115.250471	6314.025148	62.3500000

$(x_2$ or T_h), the inner radius (x_3 or R), and the length of the cylindrical section of the vessel (x_4 or L).

$$\text{Minimize: } f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (14)$$

subject to:

$$g_1(X) = -x_1 + 0.0193x_3 \leq 0, \quad (15)$$

$$g_2(X) = -x_2 + 0.00954x_3 \leq 0, \quad (16)$$

$$g_3(X) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0, \quad (17)$$

$$g_4(X) = x_4 - 240 \leq 0, \quad (18)$$

where the design variables are bounded by $x_1, x_2 \in [0, 100]$, and $x_3, x_4 \in [10, 200]$. Note that discrete variables x_1 and x_2 are multiples of 0.0625.

Table 8 compares the performance of HSO with various other algorithms for solving this problem. The results obtained by the HSO algorithm for this problem are compared with other widely used standard algorithms, which can be stated as; Aquila Optimizer (AO) [29], Arithmetic Optimization Algorithm (AOA) [30], Coati Optimization Algorithm (CA) [31], Chimp Optimization Algorithm (COA) [32], Coot algorithm (COOT) [33], Equilibrium Optimizer (EO) [34], Grey Wolf Optimizer (GWO) [21], Harris Hawks Optimizer (HHO) [35], Moth-Flame Optimization (MFO) [36], Multi-Verse Optimizer (MVO) [37], Pelican Optimization Algorithm (POA) [38], Smell Agent Optimization (SAO) [39], Sine Cosine Algorithm (SCA) [40], Slime mould algorithm (SMA) [41], Seagull Optimization Algorithm (SOA) [42], Transient Search Optimization (TSO) [43], and Whale Optimization Algorithm (WOA) [44]. The best design variables, constraints, and objective value for the considered problem found using the proposed HSO method are presented in Table 9.

The results indicate that the proposed HSO algorithm achieves one of the best solutions compared to other tested algorithms. HSO's best solution is 6059.71334, which is competitive with the top-performing algorithms such as CA, COOT, and EO. Additionally, HSO's performance in terms of mean and worst solutions is also robust, with a mean value of 6207.85426 and a worst-case scenario of 6411.11725. These figures are notably lower than those achieved by most of the other algorithms, confirming HSO's effectiveness

Table 9

The best design variables, constraints, and objective value obtained by HSO for the pressure vessel design problem.

Best solution	HSO
x1	0.7891962
x2	0.4243748
x3	42.0984456
x4	176.6365958
g1(X)	0.0000000
g2(X)	-0.0358808
g3(X)	0.0000000
g4(X)	-63.3634042
Best obj.	6059.7143350



Fig. 5. Schematic illustration of design parameters of spring.

and reliability in solving the pressure vessel design problem. The lower standard deviation 158.213319 compared to other methods also underscores HSO's consistency across multiple runs, indicating stable performance.

4.2. Optimal design of a tension/compression spring

The objective of this problem is to determine the optimal values for three continuous design variables to minimize the weight of a tension/compression spring (refer to Fig. 5). The design variables include the wire diameter (d or x_1), mean coil diameter (D or x_2), and the number of active coils (P or x_3), subject to one linear and three nonlinear inequality constraints [46]. The optimization problem is formulated as follows:

$$\min f(X) = (x_3 + 2)x_2x_1^2 \quad (19)$$

subject to the constraints:

$$g_1(X) = 1 - \frac{x_2^3x_3}{71,785x_1^4} \leq 0, \quad (20)$$

$$g_2(X) = \frac{4x_2^2 - x_1x_2}{12,566(x_1x_2 - x_1^4)} + \frac{1}{5,108x_1^2} - 1 \leq 0, \quad x_3 \in [2, 15], \quad (21)$$

$$g_3(X) = 1 - \frac{140.45x_1}{x_2x_3} \leq 0, \quad (22)$$

$$g_4(X) = \frac{x_1x_2}{1.5} - 1 \leq 0, \quad (23)$$

where the design variables are bounded by $x_1 \in [0.05, 2]$, $x_2 \in [0.25, 1.3]$, and $x_3 \in [2, 15]$.

Table 10 provides a comparison of the proposed HSO method with various algorithms for solving the spring design problem. The algorithms compared include the African Vultures Optimization Algorithm (AVOA) [47], the Bacterial Foraging Optimization Algorithm (BFOA) [48], the Cultural Algorithm (CA) [49], the Co-evolutionary Differential Evolution (CDE) [50], the Coronavirus herd immunity optimizer (CHIO) [51], the Co-evolutionary Particle Swarm Optimization (CPSO) [52], the Genetic Algorithm (GA) from two different approaches—GA(1) [53] and GA(2) [54], the Gorilla Troops Optimizer (GTO) [55], the Hybrid Genetic Algorithm (HGA) [56], the Hunger Games Search (HGS) algorithm [27], the Particle Swarm Optimization (PSO) [25], the Rat Swarm Optimizer (RSO) [57], the RUNge Kutta optimizer (RUN) [18], the Sine Cosine Algorithm (SCA) [40], Society and Civilization Optimization Algorithm (SCOA) [58], the Slime Mould Algorithm (SMA) [41], the T-Cell Algorithm (T-Cell) [59], the Whale Optimization Algorithm (WOA) [44], the War Strategy Optimization Algorithm (WSO) [60], and finally, the Young's Double-Slit Experiment Optimizer (YDSE) [46].

The results presented in Table 10 demonstrate that the proposed HSO algorithm outperforms other tested algorithms in solving the Tension/Compression design problem. HSO achieves the best solution with a value of $1.26654553\text{e}-02$, which is comparable to or better than the top-performing algorithms, such as YDSE, GTO, and PSO. Additionally, the HSO algorithm shows remarkable consistency, as evidenced by its low standard deviation of $9.11886758\text{e}-07$, indicating minimal variation across multiple runs.

Compared to other algorithms, HSO also exhibits a strong performance in terms of mean and worst-case solutions, with values of $1.26664984\text{e}-02$ and $1.26676743\text{e}-02$, respectively. These results highlight HSO's effectiveness and reliability in providing stable and optimal solution for the Tension/Compression design problem.

Table 11 presents the best design variables, constraints, and objective value obtained by the proposed HSO for this problem. The results showcase the effectiveness of the HSO algorithm in solving the Tension/Compression Spring Design problem. The HSO algorithm successfully identified optimal design variables while adhering to all the problem's constraints. The algorithm's ability to consistently find solutions that satisfy complex constraints while optimizing the objective function highlights its robustness and reliability. These results demonstrate the potential of HSO as a versatile and powerful tool for tackling challenging engineering design problems, reaffirming its applicability across a range of optimization scenarios.

Table 10

Best statistical results of various algorithms for the Tension/Compression design problem.

Methods	Best	Mean	Worst	Std.
Proposed HSO	1.26654553e-02	1.26664984e-02	1.26676743e-02	9.11886758e-07
AVOA [46]	1.2667E-02	1.3001E-02	1.3812E-02	3.8188E-04
BFOA [48]	0.012671	0.012759	N/A	1.36E-04
CA [49]	0.012721	0.013568	0.0151156	8.4E-04
CDE [50]	0.012670	0.012703	0.012790	2.07E-05
CHIO [46]	1.2670E-02	1.2881E-02	1.3454E-02	2.0080E-04
CPSO [52]	0.012674	0.012730	0.012924	5.19E-05
GA(1) [53]	0.012704	0.012769	0.012822	3.93E-05
GA(2) [54]	0.012681	0.012742	0.012973	9.5E-05
GTO [46]	1.2667E-02	1.2689E-02	1.2719E-02	1.8850E-05
HGA [56]	0.012668	0.013481	0.016155	N/A
HGS [46]	1.7249E+00	1.7813E+00	2.8528E+00	2.0988E-01
PSO [46]	1.2665E-02	1.2753E-02	1.3017E-02	9.1415E-05
RSO [46]	1.2842E-02	1.5535E+00	2.7935E+00	1.0687E+00
RUN [46]	1.2665E-02	1.3019E-02	1.7773E-02	1.0413E-03
SCA [46]	1.2738E-02	1.2901E-02	1.3067E-02	8.7128E-05
SCOA [58]	0.012669	0.012922	0.016717	5.92E-04
SMA [46]	1.2676E-02	1.2794E-02	1.3259E-02	1.7786E-04
T-Cell [59]	0.012665	0.013309	0.012732	9.4E-05
WOA [46]	1.2666E-02	1.3438E-02	1.5585E-02	9.1406E-04
WSO [46]	1.2665E-02	6.0375E-02	1.4434E+00	2.6122E-01
YDSE [46]	1.2665E-02	1.2665E-02	1.2665E-02	1.5518E-09

Table 11

The best design variables, constraints, and objective value for the Tension/Compression Spring Design problem.

Best solution	HSO
x1	0.0516063
x2	0.3547280
x3	11.4066588
g1(X)	-0.0000037
g2(X)	-0.0000024
g3(X)	-4.0498065
g4(X)	-0.7291105
Best obj.	0.0126655

4.3. Cantilever beam design optimization

The cantilever beam problem involves optimizing five hollow square sections of equal thickness, with the first section being fixed and the remaining sections allowed to move (refer to Fig. 6). The goal is to minimize the beam's weight while accounting for the vertical force applied to the free end of the beam [46]. The five sections' lengths correspond to the five decision variables (x_1, x_2, x_3, x_4, x_5), all subject to a single constraint. The cantilever beam design problem involves optimizing the decision variables, represented as $X = [x_1, x_2, x_3, x_4, x_5]$, with the objective of minimizing the cost associated with the beam. The objective function to

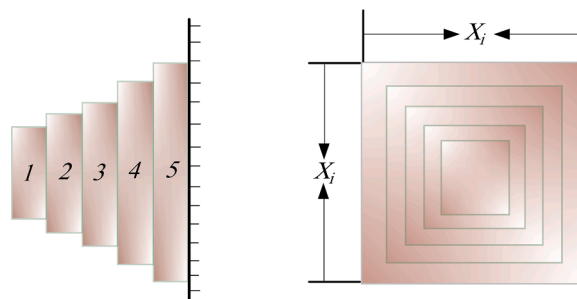
**Fig. 6.** Schematic Illustration of Design Parameters of Cantilever Beam.

Table 12

Best statistical results of various algorithms for the Cantilever Beam design problem.

Methods	Best	Mean	Worst	Std.
Proposed HSO	1.33995822	1.33996200	1.33996998	4.90684349e-06
CCS [61]	1.336526966	1.339928963	1.340058475	0.002002578
ELPSO [62]	1.34	1.34	1.34	0.00001755
GWO [65]	1.339959	1.339998	N/A	2.6519E-05
LWOA [63]	1.34	1.34101	1.349137	0.002208
MVO [65]	1.340007	1.340418	N/A	3.0554E-04
SCA [65]	1.353852	1.379948	N/A	1.6282E-02
SSA [65]	1.339957	1.339967	N/A	7.6428E-06
STOA [65]	1.340096	1.340710	N/A	7.5025E-04
TQA [65]	1.339957	1.339975	N/A	1.6258E-05
TSA [65]	1.340139	1.340901	N/A	4.3682E-04
WOA [63]	1.3435	1.3668	1.4865	0.027

be minimized is defined as:

$$\text{Minimize } f(X) = 0.0624 \times (x_1 + x_2 + x_3 + x_4 + x_5) \times L \quad (24)$$

This optimization is subject to a constraint that ensures the structural integrity of the beam:

$$g_1(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0 \quad (25)$$

The variables are bounded within the following range:

$$0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100 \quad (26)$$

The results obtained by the HSO algorithm and other competing algorithms are compared in Table 12. The competing algorithms include Convex Combination Search (CCS) [61], Enhanced Leader PSO (ELPSO) [62], Gray Wolf Optimizer (GWO) [21], Lévy Flight trajectory-based WOA (LWOA) [63], Multi Verse Optimization (MVO) [37], Sine Cosine Algorithm (SCA) [40], Salp Swarm Algorithm (SSA) [26], Sooty Tern Optimization Algorithm (STOA) [64], Termite Queen Algorithm (TQA) [65], Tunicate Swarm Algorithm (TSA) [66], and Whale Optimization Algorithm (WOA) [44].

The results presented in Table 12 highlight the superior performance of the proposed HSO algorithm for the Cantilever Beam design problem. HSO achieved the best solution with a minimum value of 1.33995822 and exhibited remarkable consistency across multiple runs, as indicated by the lowest standard deviation among all algorithms tested. The results confirm the robustness and efficiency of HSO in handling complex engineering design optimization problems.

Additionally, Table 13 presents the best design variables, constraints, and objective value obtained by the proposed HSO for this problem. The optimal design variables x_1 through x_5 have been precisely determined, yielding an objective function value of 1.3399582. The constraint $g_1(X)$ is satisfied with a negligible margin, indicating that the solution is both feasible and optimal. These results confirm the capability of HSO in achieving high-quality solutions for various engineering design problems.

4.4. Three-bar truss design optimization

The goal of this optimization problem is to identify the optimal cross-sectional areas for three bars in a truss structure to minimize the total weight of the truss. The variables x_1 , x_2 , and x_3 represent the cross-sectional areas of the bars [65]. The optimization problem is mathematically formulated as follows:

The objective function to be minimized is expressed as:

$$f(X) = (2\sqrt{2}x_1 + x_2) \times l, \quad (27)$$

Table 13

The best design variables, constraints, and objective value for the Cantilever Beam Design Optimization problem.

Best solution	HSO
x1	6.0151578
x2	5.3070154
x3	4.4956711
x4	3.5048594
x5	2.1509857
g1(X)	-0.0000025
Best obj.	1.3399582

Table 14

Best statistical results of various algorithms for the Three-bar truss design problem [65].

Methods	Best	Mean	Worst	Std.
Proposed HSO	263.895861	263.895879	263.895910	2.23913726e-05
MVO	263.895975	263.897523	N/A	1.9914E-03
GWO	263.895975	263.898476	N/A	2.2593E-03
SCA	263.921243	264.615867	N/A	3.4428E+00
SSA	263.895845	263.896421	N/A	1.0091E-03
STOA	263.897043	267.707187	N/A	7.6972E+00
TQA	263.895843	263.897510	N/A	2.6324E-03
TSA	263.896667	263.902906	N/A	5.4784E-03
WOA	263.895908	264.148125	N/A	3.4909E-01

where l denotes the length of the truss.

The optimization is subject to the following constraints:

$$g_1(X) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \quad (28)$$

$$g_2(X) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \quad (29)$$

$$g_3(X) = \frac{1}{\sqrt{2}x_1 + x_2} P - \sigma \leq 0, \quad (30)$$

with the variable range defined as $0 \leq x_1, x_2 \leq 1$. The parameters for this problem are specified as $l = 100$ cm, $P = 2 \frac{\text{KN}}{\text{cm}^2}$, and $\sigma = 2 \frac{\text{KN}}{\text{cm}^2}$.

Table 14 compares the performance of the HSO method against other competitive optimization algorithms for this problem, including Gray Wolf optimizer (GWO) [21], Multi Verse Optimization (MVO) [37], Sine Cosine Algorithm (SCA) [40], Salp Swarm Algorithm (SSA) [26], Sooty Tern Optimization Algorithm (STOA) [64], Termite queen algorithm (TQA) [65], Tunicate Swarm Algorithm (TSA) [66], and Whale Optimization Algorithm (WOA) [44].

The results presented in Table 14 demonstrate that the proposed HSO method achieves the best performance among the tested algorithms, with the lowest best, mean, and worst values, and the smallest standard deviation. This indicates a high level of consistency and robustness in the optimization process. Other algorithms such as SSA and TQA also show competitive results, but with slightly higher mean and standard deviation values, suggesting that they are less consistent compared to HSO. In contrast, algorithms like SCA and STOA exhibit larger variability, as evidenced by their higher standard deviation, indicating less reliable performance for this problem.

Moreover, Table 15 presents the best design variables, constraints, and objective value obtained by the proposed HSO for this problem. The results indicate that HSO effectively satisfies the problem constraints, with the constraint $g_1(X)$, $g_2(X)$, and $g_3(X)$ being well within acceptable limits. The best objective function value of 263.8958611 confirms the efficacy of HSO in achieving a highly optimized solution, demonstrating its capability in solving complex structural optimization problems with precision.

4.5. Welded beam design optimization

This optimization problem focuses on identifying the optimal values for four continuous decision variables to minimize the overall cost of a welded beam (refer to Fig. 7). The problem is subject to both linear and nonlinear inequality constraints. The design variables

Table 15

The best design variables, constraints, and objective value for the Three-bar Truss Design Optimization problem.

Best solution	HSO
x1	0.7887024
x2	0.4081712
g1(X)	-0.0000001
g2(X)	-1.4641893
g3(X)	-0.5358108
Best obj.	263.8958611

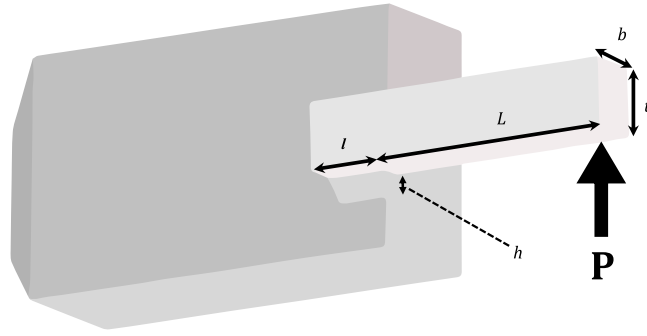


Fig. 7. Schematic Illustration of Design Parameters of Welded Beam.

x_1 , x_2 , x_3 , and x_4 correspond to the weld height (h), length (l), weld thickness (t), and breadth (b), respectively [45]. The objective function is formulated as:

$$f(X) = 1.10471x_1^2 + 0.04811x_3x_4(14 + x_2) \quad (31)$$

Subject to the following constraints:

$$g_1(X) = \tau(x) - \tau_{\max} \leq 0, \quad (32)$$

$$g_2(X) = \sigma(x) - \sigma_{\max} \leq 0, \quad (33)$$

$$g_3(X) = x_1 - x_4 \leq 0, \quad (34)$$

$$g_4(X) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0, \quad (35)$$

$$g_5(X) = 0.125 - x_1 \leq 0, \quad (36)$$

$$g_6(X) = \delta(x) - \delta_{\max} \leq 0, \quad (37)$$

$$g_7(X) = P - P_c(x) \leq 0, \quad (38)$$

where the following expressions are defined:

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \quad (39)$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}, \quad (40)$$

$$M = P\left(L + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \quad (41)$$

$$\delta(x) = \frac{4PL^3}{Ex_3^2x_4}, \quad (42)$$

$$J = 2\left[\sqrt{2}x_1x_2\left(\frac{x_1 + x_3}{2}\right)\right], \quad (43)$$

$$\sigma(x) = \frac{6PL}{x_3^2x_4}, \quad (44)$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3x_4^3}{36}}}{L^2}\sqrt{1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}}, \quad (45)$$

Given the constants: $P = 6000$ lb, $L = 14$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{\max} = 13,000$ psi, $\sigma_{\max} = 30,000$ psi, $\delta_{\max} = 0.25$ in, $x_1 \in [0.1, 2]$, $x_2 \in [0.1, 10]$, $x_3 \in [0.1, 10]$, $x_4 \in [0.1, 2]$.

Table 16

Best statistical results of various algorithms for Welded Beam design problem.

Methods	Best	Mean	Worst	Std.
Proposed HSO	1.72489555	1.72585530	1.72668977	6.92744339e-04
BFOA [48]	2.3868	2.4040	N/A	0.016
CDE [50]	1.733461	1.768158	1.824105	0.022194
COA [72]	2.1324620263	3.080523	N/A	5.44E-01
CPSO [52]	1.728024	1.748831	1.782143	0.012926
DO [72]	1.7248818489	1.726894	N/A	4.16E-03
FLA [72]	1.7625873489	2.831815	N/A	6.54E-01
FSA [69]	2.3811	2.4041	2.4889	N/A
GTO [72]	1.7248658492	1.776438	N/A	1.31E-01
GWO [65]	1.725534	1.726783	N/A	9.6817E-04
HSA-GA [70]	2.2500	2.26	2.28	0.0078
IPSO [71]	2.3810	2.3819	N/A	0.00523
KOA [72]	1.7248658492	1.724866	N/A	1.82E-15
MEA [73]	2.3816	N/A	2.38297	0.00034
MVO [65]	1.727411	1.739013	N/A	1.1024E-02
POA [72]	1.7248658492	1.724893	N/A	1.30E-04
RO [74]	1.735344	1.9083	N/A	0.173744
RUN [72]	1.7248692196	1.763343	N/A	3.40E-02
SCA [65]	1.773580	1.858889	N/A	3.6744E-02
SCA [58]	2.3854	3.2551	6.3996	0.9590
SMA [72]	1.7248973587	1.725941	N/A	1.98E-03
SSA [65]	1.728225	1.825995	N/A	9.7915E-02
SBM [75]	2.4426	2.5215	2.6315	N/A
STOA [65]	1.727887	1.737993	N/A	8.2362E-03
T-Cell [59]	2.3811	2.4398	2.7104	0.09314
TQA [65]	1.724853	1.735345	N/A	2.3349E-02
TSA [65]	1.728294	1.742316	N/A	4.9857E-03
WOA [65]	1.748147	2.259938	N/A	6.3604E-01

Table 16 provides a comparison of the proposed HSO method with various algorithms, including Artificial Gorilla Troops Optimizer (GTO) [55], Bacterial foraging optimization algorithm (BFOA) [48], Coati Optimization Algorithm (COA) [31], Co-evolutionary Differential Evolution (CDE) [50], Co-evolutionary PSO (CPSO) [52], Dandelion Optimizer (DO) [67], Fick's law optimization (FLA) [68], Filter Simulated Annealing (FSA) [69], Gray Wolf optimizer (GWO) [21], Hybrid Simulated Annealing and Genetic Algorithm (HSA-GA) [70], Improved Particle Swarm Optimizer (IPSO) [71], Kepler Optimization Algorithm (KOA) [72], Multiagent Evolutionary Algorithm (MEA) [73], Multi Verse Optimization (MVO) [37], Pelican Optimization Algorithm (POA) [38], Ray Optimization (RO) [74], RUNge Kutta Optimizer (RUN) [18], Sine Cosine Algorithm (SCA) [40], Slime Mold Algorithm (SMA) [41], Society and Civilization Optimization Algorithm (SCOA) [58], Salp Swarm Algorithm (SSA) [26], Sooty Tern Optimization Algorithm (STOA) [64], T-cell Algorithm (T-cell) [59], Termite Queen Algorithm (TQA) [65], Tunicate Swarm Algorithm (TSA) [66], and Whale Optimization Algorithm (WOA) [44].

The results presented in Table 16 indicate that the proposed HSO method achieved the best statistical performance among the tested algorithms, with the lowest value for the best solution, mean, and standard deviation. This demonstrates the superior robustness and consistency of HSO in optimizing the Welded Beam design problem. Additionally, many other algorithms, such as KOA, FLA, and SMA, also performed well, with KOA achieving a competitive best solution. However, traditional and other metaphor-based algorithms, showed higher variability and worse performance, highlighting the strength of the HSO in handling this engineering design challenge.

Furthermore, Table 17 presents the optimal solution derived from the proposed HSO algorithm. The results demonstrate the capability of the proposed HSO algorithm to find optimal solution for the Welded Beam Design problem. The table shows the specific design variable values that lead to the best objective function value of 1.7248955. The constraints $g_1(X)$ through $g_7(X)$ are all satisfied, indicating that the HSO not only identifies the optimal design variables but also ensures that all design constraints are met. This highlights the robustness and precision of HSO in handling complex engineering design problems where meeting multiple constraints is crucial for practical application.

4.6. Optimization of a speed reducer design

The objective of this problem is to minimize the weight and size of a speed reducer by optimizing seven design variables: the face width x_1 , the module of the teeth x_2 , the number of teeth on the pinion x_3 , the length of the first shaft between bearings x_4 , the length of the second shaft between bearings x_5 , the diameter of the first shaft x_6 , and the diameter of the second shaft x_7 [72]. The mathematical formulation of this optimization problem is as follows:

$$\begin{aligned} \text{Minimize } f(X) = & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ & - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned} \quad (46)$$

Table 17

The best design variables, constraints, and objective value for the Welded Beam Design problem.

Best solution	HSO
x1	0.2057278
x2	3.4706546
x3	9.0366753
x4	0.2057316
g1(X)	-0.4547542
g2(X)	-0.6234409
g3(X)	-0.0000038
g4(X)	-3.4329454
g5(X)	-0.0807278
g6(X)	-0.2355407
g7(X)	-0.1918951
Best obj.	1.7248955

Subject to the following constraints:

$$\frac{27}{x_1 x_2^2 x_3} - 1 \leq 0 \quad (47)$$

$$\frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0 \quad (48)$$

$$\frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0 \quad (49)$$

$$\frac{\sqrt{\frac{745 x_4^2}{x_2 x_3}} + 16.9 \times 10^6}{110 x_6^3} - 1 \leq 0 \quad (50)$$

$$\frac{\sqrt{\frac{745 x_5^2}{x_2 x_3}} + 157.5 \times 10^6}{85 x_7^3} - 1 \leq 0 \quad (51)$$

$$\frac{5 x_2}{x_1} - 1 \leq 0 \quad (52)$$

$$\frac{x_1}{12 x_2} - 1 \leq 0 \quad (53)$$

$$\frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0 \quad (54)$$

$$\frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0 \quad (55)$$

The design variables are bounded as follows:

$$\begin{aligned} 3.6 \geq x_1 \geq 2.6, \quad 0.8 \geq x_2 \geq 0.7, \quad 28 \geq x_3 \geq 17, \\ 8.3 \geq x_4, x_5 \geq 7.3, \quad 3.9 \geq x_6 \geq 2.9, \quad \text{and} \quad 5.5 \geq x_7 \geq 5 \end{aligned} \quad (56)$$

Here, x_1 represents the face width, x_2 is the module of the teeth, x_3 denotes the number of teeth on the pinion, x_4 is the length of the first shaft between bearings, x_5 is the length of the second shaft between bearings, x_6 indicates the diameter of the first shaft, and x_7 is the diameter of the second shaft.

Table 18 evaluates the effectiveness of the HSO approach in comparison with other leading optimization techniques for this problem, including Cooperation Search Optimization Algorithm (CSA) [76], Cuckoo search algorithm (CSA) [77], Gray Wolf optimizer (GWO) [21], Modified Differential Evolution (MDE) [78], Multi Verse Optimization (MVO) [37], Salp Swarm Algorithm (SSA) [26], Simple Constrained PSO (SC-PSO) [79], Sine Cosine Algorithm (SCA) [40], Socio-behavioural Model (SBM) Algorithm [75], Society and Civilization Optimization Algorithm (SCOA) [58], Sooty Tern Optimization Algorithm (STOA) [64], Termite queen algorithm (TQA) [65], Tunicate Swarm Algorithm (TSA) [66], and Whale Optimization Algorithm (WOA) [44].

The results presented in Table 18 indicate that the proposed HSO algorithm outperforms other optimization techniques in solving the speed reducer design problem. HSO achieves the best solution with a value of 2996.39325, demonstrating its robustness and effectiveness. The algorithm also exhibits a relatively low standard deviation, indicating consistent performance across multiple runs. Compared to other methods, such as SCA and WOA, which show higher variability and less optimal solution, HSO provides a more reliable and precise optimization approach. The performance of algorithms like CSOA and SC-PSO is also noteworthy, but HSO still maintains a superior edge in terms of both the best solution and consistency.

Table 18

Best statistical results of various algorithms for speed reducer design Problem.

Methods	Best	Mean	Worst	Std.
Proposed HSO	2996.39325	2999.48816	3007.01243	4.52548051e+00
CSA [77]	3000.981	3007.1997	N/A	4.9634
CSOA [76]	2996.34816496853	2996.348164968529	2996.3481649685305	9.90E-13
GWO [65]	2997.725189	3005.500021	N/A	4.6459E+00
MDE [78]	2996.357	2996.367	2996.390	8.20E-03
MVO [65]	3007.555169	3036.968535	N/A	1.6633E+01
SBM [75]	3008.08	3012.12	3028	N/A
SCA [65]	3050.017911	3102.237267	N/A	2.8026E+01
SC-PSO [79]	2996.348165	2996.3482	N/A	0
SCOA [58]	2994.744241	3001.758226	3009.964736	4.0091423
SSA [65]	3002.667140	3037.269055	N/A	2.4756E+01
STOA [65]	3011.906151	3028.135763	N/A	9.4705E+00
TQA [65]	2994.591855	3002.910419	N/A	5.6091E+00
TSA [65]	3017.913973	3034.726976	N/A	9.9150E+00
WOA [65]	3007.949579	3119.455059	N/A	6.9944E+01

Table 19

The best design variables, constraints, and objective value for the Speed Reducer Design Optimization problem.

Best solution	HSO
x1	3.5022378
x2	0.7000000
x3	17.0000000
x4	7.3000000
x5	7.7413480
x6	3.3504684
x7	5.2872953
g1(X)	-0.0745070
g2(X)	-0.1985110
g3(X)	-0.4993239
g4(X)	-0.9037223
g5(X)	-0.0002272
g6(X)	-0.0003586
g7(X)	-0.7025000
g8(X)	-0.0006390
g9(X)	-0.5830669
g10(X)	-0.0512736
g11(X)	-0.0032712
Best obj.	2996.3932461

Additionally, Table 19 showcases the optimal solution derived by the proposed HSO method. The results highlight the effectiveness of the HSO algorithm in solving the Speed Reducer Design Optimization problem. The optimal design variables found by HSO are consistent with a highly optimized solution, achieving a best objective value of 2996.3932461. The constraints $g_1(X)$ through $g_{11}(X)$ are all satisfied, as indicated by their negative values, which suggests that the design is not only optimal but also feasible within the specified bounds. This performance underscores HSO's capability in navigating the complex search space and ensuring constraint satisfaction while minimizing the objective function.

5. Conclusion

This paper introduced the Holistic Swarm Optimization (HSO) algorithm, a novel metaphorless optimization technique that leverages the entire population's information to inform the search process comprehensively. Unlike traditional methods that rely on partial information, HSO utilizes data from the entire population along with adaptive mutation and simulated annealing-based selection to dynamically balance exploration and exploitation, ensuring a robust and thorough search. The algorithm demonstrated competitive performance across various benchmark optimization problems, including real-world engineering design problems like the pressure vessel design, tension/compression spring design, welded beam design, cantilever beam design, and three-bar truss design optimization.

The proposed HSO algorithm's effectiveness is attributed to several key innovations, including comprehensive population utilization, effective displacement coefficients defined based on the whole population information (exploitation), adaptive mutation, and simulated annealing-based selection (exploration). These features collectively contribute to HSO's ability to avoid local optima and achieve global solutions with high reliability.

The results obtained from the experiments validate the robustness and efficiency of HSO across different optimization domains. The algorithm consistently outperformed or matched the performance of existing optimization methods, demonstrating its capability to handle complex, multimodal, and constrained optimization problems.

From a statistical perspective, the HSO algorithm exhibited superior performance in both the Friedman and Wilcoxon signed-rank tests, highlighting its statistical significance compared to other optimization techniques. These tests confirm that HSO consistently delivers better solutions or is on par with the best-performing algorithms across multiple benchmarks. The convergence behavior of HSO, as observed in the benchmark tests, shows that it not only converges in fewer iterations but also exhibits a stable and reliable convergence trend, indicating strong exploitation capabilities. Additionally, the population diversity, as measured by standard deviation, provides insight into the algorithm's ability to maintain a balance between exploration and exploitation. HSO demonstrated a well-maintained diversity throughout the optimization process, which is crucial for avoiding premature convergence and ensuring a thorough exploration of the search space.

The algorithm's performance across different problem domains demonstrates its robustness and versatility. Despite the promising performance of the HSO algorithm, it is important to acknowledge certain limitations that merit attention in future research. First, the use of global RMS-based displacement coefficients may lead to overly generalized movement dynamics in very high-dimensional or highly rugged fitness landscapes, where local exploitation is essential. Second, the algorithm's effectiveness partially relies on the diversity of the initial population; insufficient diversity can limit exploration and hinder convergence to global optima. Third, although HSO incorporates adaptive mutation and simulated annealing to prevent premature convergence, the selection of their hyperparameters (e.g., cooling rate, initial mutation rate) can influence performance in problem-specific ways. Moreover, HSO does not yet incorporate constraint-handling mechanisms tailored for highly constrained or mixed-integer problems. These limitations suggest that while HSO is robust across many benchmark scenarios, it may require additional tuning or hybridization to address extremely large-scale, constrained, or discrete optimization challenges.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT-4o, developed by OpenAI, to enhance the cohesion and coherence of the text. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

CRediT authorship contribution statement

Ebrahim Akbari: Writing – review & editing, Visualization, Software, Investigation, Data curation, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization; **Abolfazl Rahimnejad:** Writing – original draft, Validation, Methodology, Formal analysis, Writing – review & editing, Visualization, Software, Investigation, Data curation; **Stephen Andrew Gadsden:** Writing – review & editing, Resources, Investigation, Supervision, Project administration.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] G. Hu, C. Gong, X. Li, Z. Xu, CGKOA: an enhanced Kepler optimization algorithm for multi-domain optimization problems, *Comput. Methods Appl. Mech. Eng.* 425 (2024) 116964. <https://doi.org/10.1016/j.cma.2024.116964>
- [2] G. Hu, F. Huang, K. Chen, G. Wei, MNEARO: A meta swarm intelligence optimization algorithm for engineering applications, *Comput. Methods Appl. Mech. Eng.* 419 (2024) 116664. <https://doi.org/10.1016/j.cma.2023.116664>
- [3] H. Huang, R. Wu, H. Huang, J. Wei, Z. Han, L. Wen, Y. Yuan, Multi-strategy improved artificial rabbit optimization algorithm based on fusion centroid and elite guidance mechanisms, *Comput. Methods Appl. Mech. Eng.* 425 (2024) 116915. <https://doi.org/10.1016/j.cma.2024.116915>
- [4] Z. Zhang, S.-C. Chu, T.-T. Nguyen, X. Wang, J.-S. Pan, DMT-OMPA: Innovative applications of an efficient adversarial marine predators algorithm based on dynamic matrix transformation in engineering design optimization, *Comput. Methods Appl. Mech. Eng.* 431 (2024) 117247. <https://doi.org/10.1016/j.cma.2024.117247>
- [5] M. Ghasemi, E. Akbari, A. Rahimnejad, S.E. Razavi, S. Ghavidel, L. Li, Phasor particle swarm optimization: a simple and efficient variant of PSO, *Soft Comput.* 23 (19) (2019) 9701–9718. <https://doi.org/10.1007/s00500-018-3536-8>
- [6] M. Gil, E. Akbari, A. Rahimnejad, M. Ghasemi, S.A. Gadsden, Solution of optimal reactive power dispatch by Lévy-flight phasor particle swarm optimization, *Intell. Syst. Appl.* 23 (2024) 200398. <https://doi.org/10.1016/j.iswa.2024.200398>
- [7] C.E. Klein, V.C. Mariani, L. dos Santos Coelho, Cheetah based optimization algorithm: a novel swarm intelligence paradigm, in: *ESANN, Bruges, Belgium, 2018*, pp. 685–690.
- [8] Z. Zhang, Y. Li, W. Zhou, X. Chen, W. Yao, Y. Zhao, TONR: an exploration for a novel way combining neural network with topology optimization, *Comput. Methods Appl. Mech. Eng.* 386 (2021) 114083. <https://doi.org/10.1016/j.cma.2021.114083>
- [9] X. Li, B. Sun, J. Jin, J. Ding, Ship speed optimization method combining Fisher optimal segmentation principle, *Appl. Ocean Res.* 140 (2023) 103743. <https://doi.org/10.1016/j.apor.2023.103743>

- [10] L. dos Santos Coelho, V.C. Mariani, Chaotic artificial immune approach applied to economic dispatch of electric energy using thermal units, *Chaos, Solitons & Fractals* 40 (5) (2009) 2376–2383. <https://doi.org/10.1016/j.chaos.2007.10.032>
- [11] Z. Zhou, Y. Sun, X. Li, Y. Zhou, K. Tian, P. Hao, B. Wang, Shape and size optimization framework of stiffened shell using isogeometric analysis, *Thin-Walled Struct.* 204 (2024) 112311. <https://doi.org/10.1016/j.tws.2024.112311>
- [12] R. Venkata Rao, et al., BMR and BWR: Two simple metaphor-free optimization algorithms for solving real-life non-convex constrained and unconstrained problems, (2024). arXiv preprint arXiv:2407.11149.
- [13] F. Campelo, C. Aranha, Lessons from the evolutionary computation bestiary, *Artif. Life* 29 (4) (2023) 421–432. https://doi.org/10.1162/artl_a_00402
- [14] K. Sörensen, Metaheuristics—the metaphor exposed, *Int. Trans. Oper. Res.* 22 (1) (2015) 3–18. <https://doi.org/10.1111/itor.12001>
- [15] R. Rao, Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems, *Int. J. Ind. Eng. Comput.* 7 (1) (2016) 19–34. <https://doi.org/10.5267/j.ijec.2015.8.004>
- [16] R. Rao, Rao algorithms: three metaphor-less simple algorithms for solving optimization problems, *Int. J. Ind. Eng. Comput.* 11 (1) (2020) 107–130.
- [17] M. Ghasemi, A. Rahimnejad, E. Akbari, R.V. Rao, P. Trojovský, E. Trojovská, S.A. Gadsden, A new metaphor-less simple algorithm based on Rao algorithms: a fully informed search algorithm (FISA), *PeerJ Comput. Sci.* 9 (2023) e1431.
- [18] I. Ahmadianfar, A.A. Heidari, A.H. Gandomi, X. Chu, H. Chen, RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method, *Expert Syst. Appl.* 181 (2021) 115079. <https://doi.org/10.1016/j.eswa.2021.115079>
- [19] R. Singh, V.K. Pathak, A.K. Srivastava, R. Kumar, A. Sharma, A new metaphor-less optimization algorithm for synthesis of mechanisms, *Int. J. Interact. Des. Manuf. (IJIDeM)* 18 (4) (2024) 2371–2391. <https://doi.org/10.1007/s12008-023-01502-6>
- [20] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, *KanGAL Rep.* 2005005 (2005) 2005.
- [21] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* 69 (2014) 46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
- [22] E. Akbari, A. Rahimnejad, S.A. Gadsden, et al., A greedy non-hierarchical grey wolf optimizer for real-world optimization, *Electron. Lett.* 57 (13) (2021) 499–501. <https://doi.org/10.1049/ell2.12176>
- [23] A. Rahimnejad, S. Mirjalili, S.A. Gadsden, P. Trojovský, E. Trojovská, An improved hybrid whale optimization algorithm for global optimization and engineering design problems, *PeerJ Comput. Sci.* 9 (2023) e1557. <https://doi.org/10.7717/peerj-cs.1557>
- [24] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, *Computat. Intell. Lab., Zhengzhou Univ., Zhengzhou China Tech. Rep., Nanyang Technol. Univ., Singapore* 635 (2) (2013) 2014.
- [25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, 1995, pp. 1942–1948. <https://doi.org/10.1109/ICNN.1995.488968>
- [26] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, *Adv. Eng. Softw.* 114 (2017) 163–191. <https://doi.org/10.1016/j.advengsoft.2017.07.002>
- [27] P. Mehta, B.S. Yildiz, S.M. Sait, A.R. Yildiz, Mater. Test. 64 (4) (2022) 524–532. <https://doi.org/10.1515/mt-2022-0013>
- [28] T. Thaher, A. Sheta, M. Awad, M. Aldasht, Enhanced variants of crow search algorithm boosted with cooperative based island model for global optimization, *Expert Syst. Appl.* 238 (2024) 121712. <https://doi.org/10.1016/j.eswa.2023.121712>
- [29] L. Abualigah, D. Younsi, M. Abd Elaziz, A.A. Ewees, M.A.A. Al-qaness, A.H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm, *Comput. Ind. Eng.* 157 (2021) 107250. <https://doi.org/10.1016/j.cie.2021.107250>
- [30] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A.H. Gandomi, The arithmetic optimization algorithm, *Comput. Methods Appl. Mech. Eng.* 376 (2021) 113609. <https://doi.org/10.1016/j.cma.2020.113609>
- [31] M. Dehghani, Z. Montazeri, E. Trojovská, P. Trojovský, Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems, *Knowledge-Based Syst.* 259 (2023) 110011. <https://doi.org/10.1016/j.knsys.2022.110011>
- [32] M. Khishe, M.R. Mosavi, Chimp optimization algorithm, *Expert Syst. Appl.* 149 (2020) 113338. <https://doi.org/10.1016/j.eswa.2020.113338>
- [33] I. Naruei, F. Keynia, A new optimization method based on COOT bird natural life model, *Expert Syst. Appl.* 183 (2021) 115352. <https://doi.org/10.1016/j.eswa.2021.115352>
- [34] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: a novel optimization algorithm, *Knowledge-Based Syst.* 191 (2020) 105190. <https://doi.org/10.1016/j.knsys.2019.105190>
- [35] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, *Future Gener. Comput. Syst.* 97 (2019) 849–872. <https://doi.org/10.1016/j.future.2019.02.028>
- [36] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, *Knowledge-Based Syst.* 89 (2015) 228–249. <https://doi.org/10.1016/j.knsys.2015.07.006>
- [37] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, *Neural Comput. Appl.* 27 (2) (2016) 495–513. <https://doi.org/10.1007/s00521-015-1870-7>
- [38] P. Trojovský, M. Dehghani, Pelican optimization algorithm: a novel nature-inspired algorithm for engineering applications, *Sensors* 22 (3) (2022). <https://doi.org/10.3390/s22030855>
- [39] A.T. Salawudeen, M.B. Mu'azu, Y.A. Sha'aban, A.E. Adedokun, A novel smell agent optimization (SAO): an extensive CEC study and engineering application, *Knowledge-Based Syst.* 232 (2021) 107486. <https://doi.org/10.1016/j.knsys.2021.107486>
- [40] S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowledge-Based Syst.* 96 (2016) 120–133. <https://doi.org/10.1016/j.knsys.2015.12.022>
- [41] S. Li, H. Chen, M. Wang, A.A. Heidari, S. Mirjalili, Slime mould algorithm: a new method for stochastic optimization, *Future Gener. Comput. Syst.* 111 (2020) 300–323. <https://doi.org/10.1016/j.future.2020.03.055>
- [42] G. Dhiman, V. Kumar, Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems, *Knowledge-Based Syst.* 165 (2019) 169–196. <https://doi.org/10.1016/j.knsys.2018.11.024>
- [43] M.H. Qais, H.M. Hasanien, S. Alghuwainem, Transient search optimization: a new meta-heuristic optimization algorithm, *Appl. Intell.* 50 (11) (2020) 3926–3941. <https://doi.org/10.1007/s10489-020-01727-y>
- [44] S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Softw.* 95 (2016) 51–67. <https://doi.org/10.1016/j.advengsoft.2016.01.008>
- [45] E.V. Altay, O. Altay, Y. Özçevik, A comparative study of metaheuristic optimization algorithms for solving real-world engineering design problems, *CMES - Comput. Model. Eng. Sci.* 139 (1) (2023) 1039–1094. <https://doi.org/10.32604/cmcs.2023.029404>
- [46] M. Abdel-Basset, D. El-Shahat, M. Jameel, M. Abouhawwash, Young's double-slit experiment optimizer: a novel metaheuristic optimization algorithm for global and constraint optimization problems, *Comput. Methods Appl. Mech. Eng.* 403 (2023) 115652. <https://doi.org/10.1016/j.cma.2022.115652>
- [47] B. Abdollahzadeh, F.S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems, *Comput. Ind. Eng.* 158 (2021) 107408. <https://doi.org/10.1016/j.cie.2021.107408>
- [48] E. Mezura-Montes, B. Hernández-Ocana, Bacterial foraging for engineering design problems: preliminary results, in: *Memorias del 4o Congreso Nacional de Computación Evolutiva (COMCEV'2008)*, Centro de Investigación en Matemáticas Guanajuato, México, 2008.
- [49] C.A. Coello Coello, R.L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, *Eng. Optim.* 36 (2) (2004) 219–236.
- [50] F.-z. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, *Appl. Math. Comput.* 186 (1) (2007) 340–356. <https://doi.org/10.1016/j.amc.2006.07.105>
- [51] M.A. Al-Betar, Z.A.A. Alyasseri, M.A. Awadallah, I. Abu Doush, Coronavirus herd immunity optimizer (CHIO), *Neural Comput. Appl.* 33 (10) (2021) 5011–5042. <https://doi.org/10.1007/s00521-020-05296-6>
- [52] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, *Eng. Appl. Artif. Intell.* 20 (1) (2007) 89–99. <https://doi.org/10.1016/j.engappai.2006.03.003>

- [53] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, *Comput. Ind.* 41 (2000) 113–127. <https://api.semanticscholar.org/CorpusID:62129608>.
- [54] C.A. Coello Coello, E. Mezura Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Adv. Eng. Inf.* 16 (3) (2002) 193–203. [https://doi.org/10.1016/S1474-0346\(02\)00011-3](https://doi.org/10.1016/S1474-0346(02)00011-3)
- [55] B. Abdollahzadeh, F. Soleimani Gharehchopogh, S. Mirjalili, Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems, *Int. J. Intell. Syst.* 36 (10) (2021) 5887–5958. <https://doi.org/10.1002/int.22535>
- [56] H.S. Bernardino, H.J.C. Barbosa, A.C.C. Lemonge, A hybrid genetic algorithm for constrained optimization problems in mechanical engineering, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 646–653. <https://doi.org/10.1109/CEC.2007.4424532>
- [57] G. Dhiman, M. Garg, A. Nagar, V. Kumar, M. Dehghani, A novel algorithm for global optimization: rat swarm optimizer, *J. Ambient Intell. Humaniz. Comput.* 12 (8) (2021) 8457–8482. <https://doi.org/10.1007/s12652-020-02580-0>
- [58] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, *IEEE Trans. Evol. Comput.* 7 (4) (2003) 386–396. <https://doi.org/10.1109/TEVC.2003.814902>
- [59] V.S. Aragón, S.C. Esquivel, C.A.C. Coello, A modified version of a T-cell algorithm for constrained optimization problems, *Int. J. Numer. Methods Eng.* 84 (3) (2010) 351–378. <https://doi.org/10.1002/nme.2904>
- [60] T.S.L.V. Ayyarao, N.S.S. Ramakrishna, R.M. Elavarasan, N. Polumahanthi, M. Rambabu, G. Saini, B. Khan, B. Alatas, War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization, *IEEE Access* 10 (2022) 25073–25105. <https://doi.org/10.1109/ACCESS.2022.3153493>
- [61] M.A. El-Shorbagy, A.M.A. Elazeem, Convex combination search algorithm: a novel metaheuristic optimization algorithm for solving global optimization and engineering design problems, *J. Eng. Res.* (2024). <https://doi.org/10.1016/j.jer.2024.05.008>
- [62] A.R. Jordehi, Enhanced leader PSO (ELPSO): a new PSO variant for solving global optimisation problems, *Appl. Soft Comput.* 26 (2015) 401–417. <https://doi.org/10.1016/j.asoc.2014.10.026>
- [63] Y. Zhou, Y. Ling, Q. Luo, Lévy flight trajectory-based whale optimization algorithm for engineering optimization, *Eng. Comput.* 35 (7) (2018) 2406–2428.
- [64] G. Dhiman, A. Kaur, STOA: A bio-inspired based optimization algorithm for industrial engineering problems, *Eng. Appl. Artif. Intell.* 82 (2019) 148–174. <https://doi.org/10.1016/j.engappai.2019.03.021>
- [65] P. Chen, S. Zhou, Q. Zhang, N. Kasabov, A meta-inspired termite queen algorithm for global optimization and engineering design problems, *Eng. Appl. Artif. Intell.* 111 (2022) 104805. <https://doi.org/10.1016/j.engappai.2022.104805>
- [66] S. Kaur, L.K. Awasthi, A.L. Sangal, G. Dhiman, Tunicate warm aelAa new bio-inspired based metaheuristic paradigm for global optimization, *Eng. Appl. Artif. Intell.* 90 (2020) 103541. <https://doi.org/10.1016/j.engappai.2020.103541>
- [67] S. Zhao, T. Zhang, S. Ma, M. Chen, Dandelion optimizer: a nature-inspired metaheuristic algorithm for engineering applications, *Eng. Appl. Artif. Intell.* 114 (2022) 105075. <https://doi.org/10.1016/j.engappai.2022.105075>
- [68] F.A. Hashim, R.R. Mostafa, A.G. Hussien, S. Mirjalili, K.M. Sallam, Fick's law algorithm: a physical law-based algorithm for numerical optimization, *Knowledge-Based Syst.* 260 (2023) 110146. <https://doi.org/10.1016/j.knosys.2022.110146>
- [69] A.-R. Hedar, M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, *J. Global Optim.* 35 (4) (2006) 521–549. <https://doi.org/10.1007/s10898-005-3693-z>
- [70] S.-F. Hwang, R.-S. He, A hybrid real-parameter genetic algorithm for function optimization, *Adv. Eng. Inf.* 20 (1) (2006) 7–21. <https://doi.org/10.1016/j.aei.2005.09.001>
- [71] S. He, E. Prempan, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, *Eng. Optim.* 36 (5) (2004) 585–605.
- [72] M. Abdel-Basset, R. Mohamed, S.A.A. Azeem, M. Jameel, M. Abouhawwash, Kepler optimization algorithm: a new metaheuristic algorithm inspired by Kepler's laws of planetary motion, *Knowledge-Based Syst.* 268 (2023) 110454. <https://doi.org/10.1016/j.knosys.2023.110454>
- [73] J. Zhang, C. Liang, Y. Huang, J. Wu, S. Yang, An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization, *Appl. Math. Comput.* 211 (2) (2009) 392–416. <https://doi.org/10.1016/j.amc.2009.01.048>
- [74] A. Kaveh, M. Khayatizadeh, A new meta-heuristic method: ray optimization, *Comput. Struct.* 112–113 (2012) 283–294. <https://doi.org/10.1016/j.compstruc.2012.09.003>
- [75] S. Akhtar, K. Tai, T. Ray, A socio-behavioural simulation model for engineering design optimization, *Eng. Optim.* 34 (4) (2002) 341–354.
- [76] Z.-k. Feng, W.-j. Niu, S. Liu, Cooperation search algorithm: a novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems, *Appl. Soft Comput.* 98 (2021) 106734. <https://doi.org/10.1016/j.asoc.2020.106734>
- [77] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, *Eng. Comput.* 29 (1) (2013) 17–35. <https://doi.org/10.1007/s00366-011-0241-y>
- [78] E. Mezura-Montes, C.A. Coello Coello, J. Velázquez-Reyes, L. Muñoz-Dávila, Multiple trial vectors in differential evolution for engineering design, *Eng. Optim.* 39 (5) (2007) 567–589.
- [79] L.C. Cagnina, S.C. Esquivel, C.A.C. Coello, Solving engineering optimization problems with the simple constrained particle swarm optimizer, *Informatica* 32 (3) (2008) 319–326.