Cluster Computing (2025) 28:853
https://doi.org/10.1007/s10586-025-05543-2

Dholes-inspired optimization (DIO): a nature-inspired algorithm for engineering optimization problems

Ali El Romeh[1] · **Václav Snášel**[2] · **Seyedali Mirjalili**[1,2,3]

Received: 27 January 2025 / Revised: 5 May 2025 / Accepted: 12 May 2025

Abstract

This paper proposes the Dhole-Inspired Optimization (DIO) algorithm, a novel metaheuristic inspired by the cooperative hunting behavior of dholes (*Cuon alpinus*). The algorithm employs a hierarchical pack structure, where a Lead Vocalizer guides the search process while subordinate members adapt their movements to balance exploration and exploitation dynamically. This structure enhances search efficiency, prevents premature convergence, and improves solution accuracy across different problem landscapes. DIO is benchmarked on unimodal, multimodal, and composite test functions, demonstrating superior performance compared to established optimization algorithms, including Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Gravitational Search Algorithm (GSA), Fast Evolutionary Programming (FEP), and Differential Evolution (DE). The results show that DIO achieves higher accuracy and faster convergence rates on a majority of test cases, validating its robustness and reliability in tackling complex optimization problems. Furthermore, the algorithm is evaluated on real-world engineering applications, demonstrating its adaptability and effectiveness in practical scenarios. The findings highlight DIO as a versatile and competitive optimization approach, suitable for a wide range of applications in science and engineering.

Keywords Optimization algorithms · Nature-inspired algorithms · Cooperative hunting · Swarm intelligence · Metaheuristics · High-dimensional optimization · Artificial Inteligence

1 Introduction

Optimization problems are fundamental to numerous scientific and engineering disciplines, encompassing applications in resource allocation, network design, machine learning, and financial modeling. These problems typically require finding an optimal solution within a large, often complex, search space that is constrained by various objectives and uncertainties [1]. Traditional optimization techniques, including linear, integer, and dynamic programming, are well-suited for structured, low-dimensional problems but struggle in scenarios involving non-linear, non-convex, or multi-modal landscapes [2]. These challenges arise due to the presence of numerous local optima, discontinuities, and high-dimensional decision spaces, necessitating the development of adaptive and scalable optimization techniques.

In response to these limitations, nature-inspired metaheuristic algorithms have gained prominence. These approaches mimic biological processes such as evolution, swarm intelligence, and social interactions, demonstrating superior robustness and adaptability in navigating complex search spaces [3, 4]. Unlike classical deterministic methods, which follow a fixed sequence of operations and are effective only for smooth, differentiable functions [5], stochastic metaheuristics introduce randomness to escape local optima and ensure broader exploration [6].

✉ Ali El Romeh
ali.elromeh@torrens.edu.au

✉ Seyedali Mirjalili
ali.mirjalili@torrens.edu.au

Václav Snášel
vaclav.snasel@vsb.cz

[1] Centre for Artificial Intelligence Research and Optimization, Torrens University Australia, Melbourne 3000, Australia

[2] Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava 708 33, Czech Republic

[3] University Research and Innovation Center, Obuda University, Budapest 1034, Hungary

Published online: 19 September 2025

Among stochastic approaches, population-based algorithms have been widely adopted due to their ability to maintain a diverse set of candidate solutions, improving the trade-off between exploration and exploitation. Genetic Algorithms (GAs) [7] leverage selection, crossover, and mutation operators to evolve populations toward optimal solutions. Similarly, Particle Swarm Optimization (PSO) [8] simulates collective intelligence observed in flocks of birds or schools of fish, adjusting particle positions based on both individual and swarm-wide experiences. Swarm Intelligence (SI) techniques, including Ant Colony Optimization (ACO) [9] and Artificial Bee Colony (ABC) [10], further extend these concepts by emulating pheromone-based foraging and cooperative search strategies [11].

Metaheuristic optimization frameworks, such as Simulated Annealing (SA) [12] and Tabu Search [13], introduce probabilistic mechanisms and memory-based strategies to refine search trajectories. Extensions of these methods, including Differential Evolution (DE) [14] and Genetic Programming (GP) [15], further enhance adaptability across various problem domains [16, 17]. Despite their success, existing approaches still face challenges such as premature convergence, inefficient balance between exploration and exploitation, and limited generalizability to high-dimensional optimization landscapes [18].

Recent advances in nature-inspired algorithms have sought to address these issues by introducing novel search strategies. Algorithms such as the Grey Wolf Optimizer (GWO) [19] and Salp Swarm Algorithm (SSA) [20] model predator-prey interactions to achieve dynamic search control. Other recent developments, including the Draco Lizard Optimizer (DLO) [21], Eurasian Lynx Optimizer (ELO) [22], and Artificial Meerkat Algorithm (AMA) [23], incorporate adaptive leadership mechanisms and cooperative strategies to improve convergence behavior. However, many newly proposed methods rely on modifications of existing frameworks rather than fundamentally novel search paradigms, limiting their contributions to incremental improvements rather than groundbreaking innovations [24].

While the expansion of metaheuristic algorithms continues, concerns remain regarding redundancy and benchmarking inconsistencies in evolutionary computation [18]. Many recent approaches either introduce minor parameter variations or adapt well-known techniques without providing a distinct theoretical foundation. Furthermore, the ongoing debate on whether manually designed metaheuristics can still outperform automated algorithm design underscores the necessity of biologically meaningful, structured search mechanisms that enhance robustness and adaptability [24].

To address these challenges, this paper introduces the DIO algorithm, a novel metaheuristic inspired by the cooperative hunting strategies and social hierarchy of dholes (*Cuon alpinus*). Unlike existing swarm-based algorithms such as PSO [8] and GWO [19], DIO employs a multi-stage leadership model where the search process is guided by a dynamically selected lead vocalizer. This structured hierarchy enables adaptive balancing between exploration and exploitation, a key factor in solving complex optimization problems efficiently. Additionally, DIO incorporates a vocalization-based guidance mechanism, wherein the influence of the lead vocalizer gradually decreases over iterations, differentiating it from purely random exploration seen in other bio-inspired approaches.

By modeling the hierarchical social structure of dhole packs, DIO enhances information exchange across candidate solutions and mitigates premature convergence. Unlike single-leader paradigms found in GWO, DIO enables multi-stage leadership transitions, facilitating a more diverse search across the solution space. Furthermore, a territorial behavior mechanism prevents rapid convergence by enforcing search diversity and confining the optimization process within feasible regions.

Experimental evaluations demonstrate that DIO outperforms existing metaheuristics across benchmark functions, achieving superior convergence speed and solution quality compared to PSO, GWO, and DE. In addition to benchmark tests, DIO was also applied to real-world engineering optimization problems, including the Welded Beam Design, and Pressure Vessel Design problems, demonstrating its practical applicability. This paper contributes to the ongoing evolution of metaheuristic optimization by introducing a biologically meaningful, structured search framework that extends beyond incremental modifications of existing algorithms. The remainder of this paper is organized as follows: Sect. 2 reviews related work on metaheuristic optimization, Sect. 3 details the mathematical formulation and inspiration behind the DIO algorithm, Sect. 4 presents experimental results, Sect. 5 discusses the application of DIO to classical engineering problems, and Sect. 6 concludes with future research directions.

The source code of the proposed Dhole-Inspired Optimization (DIO) algorithm is publicly available at MathWorks File Exchange and GitHub to facilitate reproducibility and future research.

•MathWorks: https://au.mathworks.com/matlabcentral/fileexchange/181141-dholes-inspired-optimization-dio
•GitHub: https://github.com/Alyromeh/Dholes-Inspired-Optimization-DIO

2 Literature review

2.1 Single-objective optimization problems

Single-objective optimization deals with improving a single performance criterion, which usually involves either maximizing or minimizing an objective function. This function represents the goal of the problem at hand, like minimizing production costs in manufacturing or increasing the yield in agricultural production. A common way to represent such problems is through the following mathematical formulation, where $f(\mathbf{x})$ denotes the objective function $f : \mathbb{R}^n \to \mathbb{R}$, and $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}$ is the vector of decision variables:

$$\begin{aligned} \text{Optimize:} \quad & f(\mathbf{x}) \\ \text{Subject to:} \quad & g_j(\mathbf{x}) \geq 0, \qquad j = 1, 2, \ldots, m \\ & h_k(\mathbf{x}) = 0, \qquad k = 1, 2, \ldots, p \\ & l_i \leq x_i \leq u_i, \quad i = 1, 2, \ldots, n \end{aligned}$$

Here:

- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are the decision variables that need to be determined for the objective to be optimized.
- $f(\mathbf{x})$ is the objective function, which serves as a mathematical expression of the performance criterion.
- $g_j(\mathbf{x}) \geq 0$ represents the *j*-th inequality constraint, ensuring certain conditions are met by the solution.
- $h_k(\mathbf{x}) = 0$ is the *k*-th equality constraint, imposing strict conditions that the solution must satisfy.
- l_i and u_i are the lower and upper bounds for the *i*-th decision variable, defining permissible values.

The difficulty in solving single-objective optimization problems stems from various factors, including the nature of the search space, the constraints involved, local optima, and the need for efficient convergence methods.

One major challenge is the high dimensionality of many problems. As the number of decision variables *n* grows, the search space expands exponentially. This phenomenon, often referred to as the "curse of dimensionality," makes finding the optimal solution more difficult because the number of possible solutions increases rapidly, and the search space becomes more complex. To tackle such large problems, algorithms like evolutionary algorithms are often used, as they are adept at exploring large spaces while avoiding being trapped in local optima by maintaining diverse solutions [25].

Whether the search space is continuous or discrete also influences which optimization techniques are suitable. In continuous search spaces, gradient-based methods are frequently used, as they rely on the gradient of the objective function to guide the search process. However, for discrete search spaces, where variables can only take distinct values, techniques like integer programming or combinatorial optimization are more appropriate. Discrete spaces often pose challenges because they are typically non-convex, requiring methods capable of effectively exploring the fragmented search space without relying on gradient information.

Constraints significantly shape the search space by defining the feasible region, i.e., the set of solutions that satisfy the problem's conditions. Inequality constraints $g_j(X)$ create boundaries within this region, while equality constraints $h_k(X)$ restrict the solution to specific areas within it. In many real-world applications, such as structural optimization, constraints can be highly nonlinear and create fragmented feasible regions, making large portions of the search space unusable. Effective optimization algorithms must be able to navigate around these boundaries. Penalty methods, which impose a cost on violating constraints, are often employed to guide the search back into the feasible region. Alternatively, repair methods adjust infeasible solutions to make them valid within the constraints [26].

A common obstacle in optimization is the existence of local optima-solutions that are superior to nearby alternatives but not globally optimal. Algorithms can easily become stuck in these local optima, leading to suboptimal results. This issue is particularly problematic in complex, multimodal landscapes where objective functions have multiple peaks and valleys. Global optimization methods like simulated annealing help address this by occasionally accepting worse solutions to escape local optima. Evolutionary algorithms also counteract this issue by maintaining diverse populations of solutions, allowing exploration of multiple regions of the search space at once. Genetic diversity is especially important for avoiding premature convergence on local optima, promoting exploration of new areas [27].

Finally, the efficiency of optimization algorithms is often measured by how quickly they converge to an optimal or near-optimal solution. Fast convergence can be beneficial, but it also increases the risk of premature convergence to suboptimal solutions, especially in complex landscapes with many local optima. Hence, a balance must be struck between exploration-searching new regions of the search space-and exploitation-refining known good solutions. Adaptive algorithms, like Differential Evolution, adjust this balance dynamically to enhance convergence without sacrificing solution quality [28]. These algorithms often use mechanisms such as adaptive parameter control, where parameters are adjusted on-the-fly based on the progress of the optimization process.

As single-objective optimization problems become increasingly complex due to advances in technology and the growing size of datasets, there is a continual demand for

more sophisticated algorithms and optimization techniques. The need to address high-dimensional search spaces, intricate constraints, and the prevalence of local optima requires innovative solutions that can balance exploration and exploitation efficiently. Future trends are likely to focus on hybrid methods that combine various optimization strategies, leveraging machine learning and adaptive algorithms to tackle these challenges more effectively. These advancements in optimization algorithms form the basis for the exploration of various single-objective optimization algorithms, which are discussed in the following section.

2.2 Single-objective optimization algorithms

Single-objective optimization algorithms play a vital role in solving problems where one main performance criterion, such as minimizing cost or maximizing efficiency, is the focus. Broadly speaking, these algorithms fall into two main categories: deterministic and stochastic methods. Each category brings its own set of strengths and limitations, and their effectiveness depends heavily on the specific nature of the problem at hand.

Deterministic algorithms are appreciated for their predictability and consistency. They will always provide the same solution if started with the same initial conditions, making them highly reliable in scenarios where the search space is smooth and well-defined. Methods like Gradient Descent and Newton's Method, often used in engineering and machine learning, fall under this category. These algorithms work by progressively moving toward the optimal solution based on the gradient of the objective function. However, one major downside of deterministic methods is their tendency to get stuck in local optima. Since they follow a fixed path dictated by the gradient, they can struggle in more complex environments where the objective function has many peaks and valleys. This limitation is particularly pronounced in tasks such as neural network training, where the optimization landscape is highly non-convex and irregular [29].

On the other hand, stochastic algorithms introduce an element of randomness into the search process, giving them the ability to explore the search space more freely. This randomness allows stochastic algorithms to escape local optima more easily, enhancing their chances of finding the global optimum. However, this flexibility comes at the cost of consistency; two runs of the same algorithm with identical starting conditions may yield different results. To improve the reliability of these methods, various strategies are employed, such as fine-tuning algorithm parameters, increasing the number of iterations, or performing multiple runs. These approaches can help reduce variability and enhance overall performance.

Stochastic algorithms can be further divided into individualist and collective approaches. Individualist algorithms focus on refining a single solution over time. Tabu Search, for example, avoids revisiting previously explored solutions by using a memory structure, allowing the algorithm to move beyond local optima and explore new areas [30]. Similarly, Hill Climbing starts with an initial solution and makes incremental changes to improve it, selecting the best neighboring solution at each step [29]. Iterated Local Search (ILS) builds on this idea by repeatedly modifying the solution, applying local search to each new version, helping the algorithm to escape local optima and explore more promising regions of the search space [31]. Simulated Annealing (SA), inspired by the cooling process of metals, allows the algorithm to occasionally accept worse solutions in the short term to avoid getting stuck in local optima. As the search progresses, the likelihood of accepting these worse solutions decreases over time, mimicking the cooling of a metal [12]. These individualist methods are generally efficient and require fewer function evaluations, making them ideal for simpler problems or resource-constrained environments. However, their focus on a single solution can make them vulnerable to becoming trapped in local optima when faced with more complex landscapes.

In contrast, collective stochastic algorithms involve multiple solutions evolving together, which makes them more robust in the face of local optima. Genetic Algorithms (GA), for instance, mimic the process of natural selection, evolving a population of solutions over several generations [7]. This collective approach allows for exploration of a broader search space, as the diversity of the population helps avoid premature convergence. Particle Swarm Optimization (PSO) takes inspiration from the social behavior of birds flocking or fish schooling. Each particle in PSO represents a potential solution and adjusts its position based on its own experience and the successes of its neighbors [8]. Similarly, Ant Colony Optimization (ACO) simulates the foraging behavior of ants, where artificial ants construct solutions by traversing a graph and laying down pheromones to guide other ants toward promising areas of the search space [32]. Differential Evolution (DE) maintains a population of candidate solutions and continuously combines and mutates them to explore the search space, making it particularly effective for optimizing real-valued functions [28]. These collective approaches are particularly well-suited to complex and high-dimensional optimization problems where the likelihood of getting stuck in local optima is high. By maintaining a diverse set of solutions and encouraging collaboration between them, these algorithms can thoroughly explore the search space, reducing the chances of getting stuck and increasing the likelihood of finding the global optimum.

One of the ongoing challenges with both individualist and collective optimization algorithms is finding the right balance between exploration and exploitation. Exploration involves searching through new areas of the solution space, while exploitation focuses on refining and improving known good solutions. If the algorithm shifts to exploitation too early, it may miss out on better solutions located elsewhere in the search space. On the other hand, excessive exploration can waste valuable computational resources and delay convergence. Adaptive algorithms, such as Differential Evolution, dynamically adjust this balance to optimize performance [28]. In Genetic Algorithms, for example, this balance is managed by tuning the mutation rate and crossover probability, while in Simulated Annealing, the cooling schedule determines how gradually the algorithm transitions from exploring new solutions to exploiting the best-found ones. The design of these adaptive mechanisms remains an area of active research, as complex optimization problems demand more sophisticated strategies to maintain the right balance between exploration and exploitation.

The increasing popularity of collective stochastic algorithms has spurred interest in developing new optimization techniques inspired by natural, physical, and human processes. Swarm-based algorithms, such as Particle Swarm Optimization (PSO) [8], Ant Colony Optimization (ACO) [32], and Artificial Bee Colony (ABC) [10], have been successfully applied to a wide range of optimization problems, from engineering design to machine learning [33, 34]. Numerous studies have also explored enhanced variants and hybridizations of these swarm-based methods to further improve convergence and adaptability [35–38]. Physics-based algorithms, such as the Gravitational Search Algorithm (GSA) [27], Charged System Search (CSS) [39], and Central Force Optimization (CFO) [40], simulate the interactions of particles or bodies under physical forces, making them particularly suitable for problems where the optimization landscape mirrors a physical system. Evolutionary algorithms, such as Biogeography-Based Optimization (BBO) [26], Evolution Strategy (ES) [41], and Differential Evolution (DE) [28], continue to be widely used due to their flexibility and effectiveness in addressing complex optimization challenges. Additionally, human-based algorithms, inspired by human learning and decision-making processes, such as Teaching and Learning Based Optimization (TLBO) [42], Seeker Optimization Algorithm (SOA) [43], Soccer League Competition (SLC) [44], and Mine Blast Algorithm (MBA) [45], have proven effective in tackling problems that require a high degree of adaptability and flexibility.

Recent advances in metaheuristic optimization have introduced several novel nature-inspired algorithms that incorporate unique behavioral, biological, and physical mechanisms. The Artificial Lemming Algorithm (ALA) [46] models the risk-taking migratory behaviors of lemmings, allowing solutions to balance exploration and exploitation dynamically. The Multi-Strategy Boosted Snow Ablation Optimizer (MSAO) [47] integrates multiple search strategies to improve convergence and solution quality, particularly in complex, multimodal landscapes. Additionally, the Improved Hybrid Aquila Optimizer and African Vultures Optimization Algorithm (IHAOAVOA) [48] combines elements from two powerful nature-inspired algorithms, utilizing composite opposition-based learning and a fitness-distance balance mechanism to refine the optimization process.

Beyond biological inspirations, recent developments in optimization also include physics-based and behavioral models. The Artificial Satellite Search (ASS) [49] models the orbital motion of satellites, integrating gravitational attraction and orbital adjustments to refine search movements. The Crested Ibis Algorithm (CIA) [50] simulates the migratory behavior of crested ibis birds, employing flocking strategies to navigate the search space efficiently. The Starfish Optimization Algorithm (SFOA) [51] utilizes the regenerative properties of starfish to adapt search strategies dynamically, enhancing exploration and exploitation balance. Similarly, the Fungal Growth Optimizer (FGO) [52] mimics the hyphal growth and spore dispersal mechanisms in fungi to create a self-adaptive search process.

Additionally, the Multiplayer Battle Game Optimizer (MBGO) [53] incorporates tactical decision-making inspired by competitive multiplayer games, leveraging attack-defense mechanisms to improve search efficiency. These recently proposed algorithms further highlight the growing interest in bio-inspired and physics-driven metaheuristics. While each of these techniques brings unique characteristics to the field, their effectiveness is dependent on the nature of the optimization problem.

The DIO algorithm builds upon these advancements by incorporating hierarchical leadership dynamics and cooperative hunting strategies, distinguishing itself from conventional swarm-based and population-based approaches. Its vocalization-based guidance mechanism offers a novel alternative to managing exploration and exploitation, contributing to the continuous development of adaptive and self-organizing optimization techniques.

Among the emerging nature-inspired algorithms, the Aquila Optimizer (AFO) [54] and Harris Hawks Optimization (HHO) [55] have received considerable attention. AFO mimics the hunting strategies of aquila (eagle) birds through soaring, gliding, and sudden dives, effectively balancing exploration and exploitation phases. It has shown promising results in solving complex engineering problems and high-dimensional tasks. On the other hand, HHO models the cooperative hunting behavior of Harris hawks, alternating

between soft and hard besiege strategies based on prey escape energy. This dynamic switching mechanism makes HHO well-suited for global optimization problems characterized by rugged or deceptive search landscapes. Both AFO and HHO contribute to the growing family of bio-inspired metaheuristics, offering flexible and powerful frameworks for tackling diverse optimization challenges.

2.3 Nature-inspired optimization algorithms: a comparative analysis

The field of nature-inspired optimization algorithms is rich with diverse strategies tailored to various complex problems. Among these, the DIO algorithm presents a novel approach by incorporating adaptive learning and dynamic behavior adjustments inspired by the strategic hunting behaviors of Dholes. This mechanism is designed to enhance flexibility and robustness in changing environments, which contrasts with the Dholes Hunting (DhoH) algorithm by Nguyen et al. [56]. While DhoH focuses on gradient approximation for blockchain consensus optimization, emphasizing precision and efficiency within a specific domain, DIO extends the application range of Dholes-inspired algorithms to broader optimization challenges.

Unlike DhoH, which is specialized for blockchain technologies, DIO is developed to adapt dynamically to a variety of complex scenarios beyond a single domain, showcasing its capability to handle unpredictability and fluctuating problem spaces effectively. This adaptability makes DIO particularly suitable for applications where environmental conditions or problem parameters are not constant or are prone to change, offering a robust solution in scenarios where traditional methods might struggle.

By leveraging the strengths of collective intelligence and environmental interaction, DIO not only matches traditional algorithms in accuracy and efficiency but also excels in environments that demand high adaptability, thus broadening the scope and applicability of nature-inspired algorithms in tackling diverse and dynamic optimization problems.

In summary, single-objective optimization algorithms are diverse and encompass a wide range of methods, from deterministic approaches that provide reliability and predictability to stochastic methods that offer flexibility and robustness. While deterministic methods are effective for simpler problems, their susceptibility to local optima limits their application in complex scenarios. Stochastic algorithms, particularly those employing collective strategies, are more capable of handling complex optimization problems with high dimensionality and multiple local optima. The ongoing development of adaptive mechanisms to balance exploration and exploitation phases is crucial to advance these algorithms. As the field continues to evolve, the integration of new biological, physical, and human-inspired approaches promises to enhance the capabilities of optimization algorithms, addressing increasingly complex and dynamic problems across various domains.

3 Inspiration for dholes-inspired optimization (DIO) algorithm

The dhole (*Cuon alpinus*), commonly known as the Asiatic wild dog, is a highly social member of the Canidae family, primarily found in the forests and grasslands of Central, East, and Southeast Asia [57–59]. Dholes are notable for their complex social structures [60], cooperative hunting behaviors [61–63], and sophisticated vocal communication systems [64, 65]. These attributes have been studied extensively [66], revealing highly efficient strategies for resource allocation, coordination, and adaptability, which closely parallel the principles of optimization algorithms [67, 68]. Their evolutionary lineage and genetic patterns have also been explored to understand adaptive behavior [69]. The DIO algorithm translates these natural behaviors into computational strategies, offering a novel metaheuristic approach for solving complex optimization problems.

The DIO algorithm draws upon the dhole's social dynamics, adaptability, and cooperative behavior to address the challenges inherent in multi-modal and high-dimensional optimization landscapes. This bio-inspired method introduces new mechanisms for balancing exploration and exploitation, promoting robust convergence while maintaining diversity within the solution space. By leveraging the adaptive and cooperative behaviors observed in dholes, DIO is designed to navigate rugged and discontinuous search landscapes more effectively than traditional methods [70, 71]. The following subsections detail the core components of the DIO algorithm, each modeled after specific dhole behaviors.

3.1 Lead vocalizer and social structure

Dholes live in tightly-knit packs, ranging from small family units to large groups of up to 40 individuals. Central to each pack's organization is a Lead Vocalizer, who directs the group's movements and coordinates hunting efforts through vocal signals. Unlike typical dominance-based leadership seen in other species, the Lead Vocalizer's role relies on guiding and coordinating the pack rather than asserting physical dominance. This concept forms the foundation of the DIO algorithm, where the Lead Vocalizer represents the current best solution.

In DIO, the Lead Vocalizer's position in the search space acts as the focal point, guiding other candidate solutions

towards optimal regions. The algorithm adaptively balances global exploration and local exploitation by dynamically adjusting the influence of the Lead Vocalizer, ensuring that the search remains diverse and does not prematurely converge to suboptimal solutions. This hierarchical approach differentiates DIO from other swarm-based algorithms by incorporating a flexible leadership mechanism that adjusts based on the progress of the search.

3.2 Vocal communication and adaptive control

One of the defining characteristics of dholes is their use of diverse vocalizations, which play a crucial role in coordinating pack movements during hunts. These vocal signals are used to communicate the position of prey, direct group members, and respond to environmental changes in real-time. In the DIO algorithm, this vocal communication is translated into an adaptive control mechanism that governs the influence of the Lead Vocalizer over time.

Initially, the Lead Vocalizer exerts a strong influence, guiding the search process toward promising regions of the solution space. As the algorithm progresses, this influence diminishes, allowing for a broader exploration of the search space and reducing the risk of premature convergence. This dynamic adjustment ensures that the algorithm retains its capacity for global search while honing in on the best solutions as the search space is explored. This adaptive mechanism represents a novel contribution to metaheuristic optimization, offering a more flexible and responsive approach to balancing exploration and exploitation.

3.3 Cooperative hunting and solution adjustment

Dholes are renowned for their cooperative hunting strategies, where pack members work together to outmaneuver and capture prey. The pack disperses to encircle the target, with some individuals driving the prey towards others lying in ambush. This high level of coordination and adaptability allows the pack to adjust its strategy in real-time based on the movements of the prey and the positions of other pack members.

The DIO algorithm mimics this cooperative behavior by ensuring that candidate solutions (analogous to individual dholes) adjust their positions in the search space based on the Lead Vocalizer's guidance and the positions of neighboring solutions. This cooperative interaction between solutions fosters a more efficient exploration of the search space, enabling the algorithm to adapt dynamically as the optimization process unfolds. The cooperative adjustment mechanism allows DIO to maintain solution diversity while preventing stagnation, making it particularly effective in navigating complex, multi-modal landscapes.

3.4 Territorial instincts and boundary constraints

In addition to their cooperative behaviors, dholes exhibit strong territorial instincts, defending their hunting grounds from intruders and ensuring the safety of the pack. This territorial behavior ensures that the pack operates within a defined space, preventing individuals from straying too far and minimizing unnecessary risk. In the context of optimization, this behavior is translated into boundary constraints that confine the solutions within the feasible region of the search space.

The DIO algorithm enforces these boundary constraints to ensure that candidate solutions remain within the defined search space, preventing divergence and promoting a more focused search for the global optimum. By maintaining the integrity of the search space and implementing adaptive boundary checks, the algorithm enhances its efficiency in finding high-quality solutions while minimizing the risk of exploring infeasible regions. This boundary enforcement mechanism ensures that the search remains effective across a wide range of optimization problems, further distinguishing DIO from other metaheuristic methods.

These features make the DIO algorithm a powerful tool for solving complex, high-dimensional optimization problems, particularly in scenarios where existing algorithms struggle to balance exploration and exploitation effectively. By incorporating biological strategies into a computational framework, DIO offers a novel and adaptable approach to tackling some of the most challenging optimization tasks in engineering and data science.

3.5 Mathematical framework and algorithm

Building upon the intricate social behaviors of dholes, the DIO algorithm translates their vocal communication, cooperative hunting strategies, and territorial instincts into a robust metaheuristic framework for solving complex optimization problems. The workflow of the DIO algorithm is illustrated in Fig. 1, detailing the sequence of operations from initialization to convergence. The algorithm begins by randomly initializing a population of candidate solutions, represented as $\vec{D}$, each corresponding to a dhole within the search space. The best-performing candidate in each iteration assumes the role of the Lead Vocalizer, guiding the optimization process. The hierarchical leadership mechanism allows for an adaptive balance between exploration and exploitation, ensuring that the search process does not stagnate in local optima. To prevent excessive divergence and maintain search efficiency, territorial constraints are applied, ensuring that candidate solutions remain within feasible boundaries.

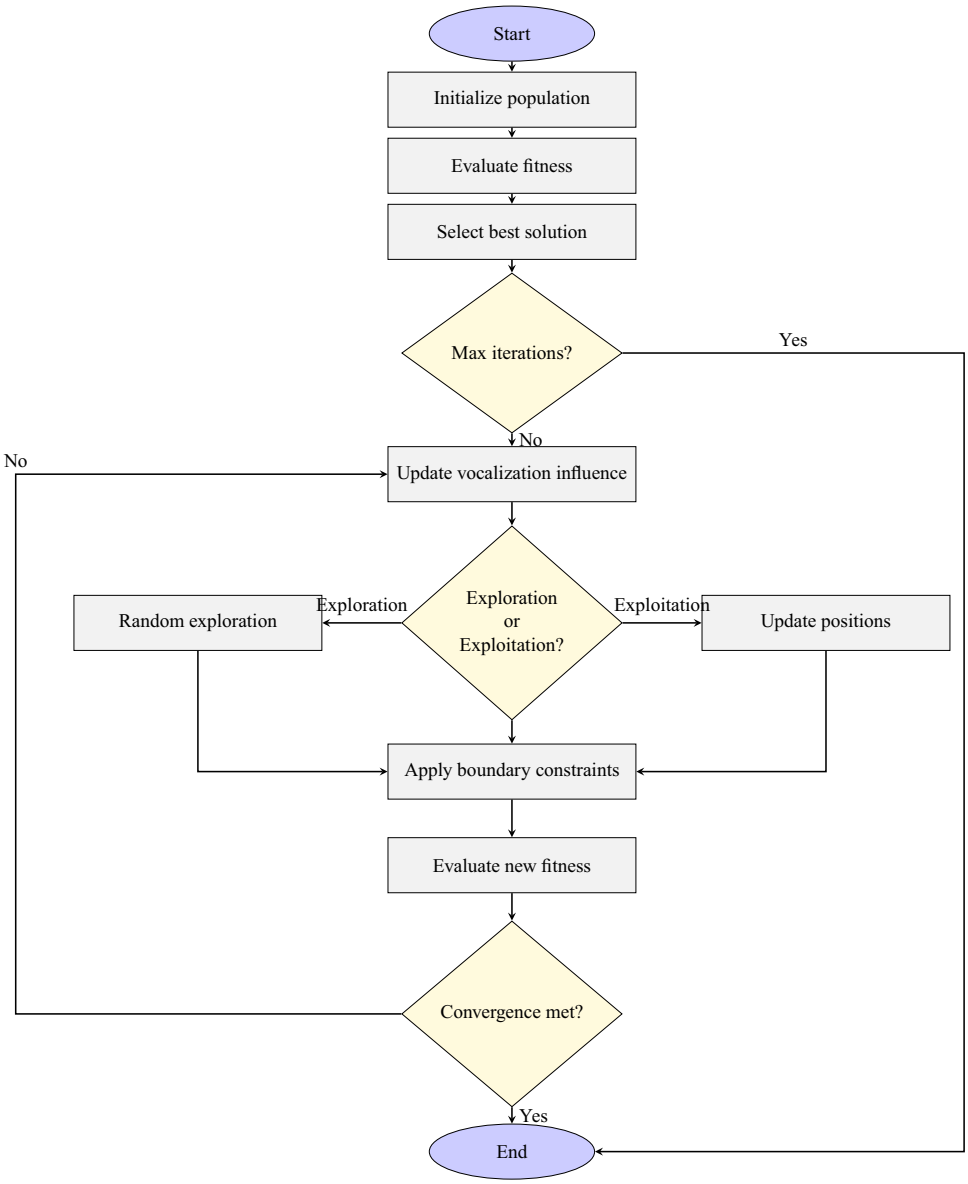


Fig. 1 Dholes-inspired optimization (DIO) flowchart

The algorithm initializes a population of candidate solutions ($\vec{D}$), each representing a dhole, and iteratively refines their positions based on vocalization influence, cooperative adjustments, and territorial constraints. The most optimal candidate solution in each iteration assumes the role of the Lead Vocalizer, guiding the optimization process. Leadership transitions, inspired by the hierarchical structure of dhole packs, allow for an adaptive balance between exploration and exploitation. The mathematical formulations of these components are detailed as follows.

3.5.1 Vocal communication modeling

Dholes rely on vocal communication to coordinate movements, particularly during hunting. In the DIO algorithm, the influence of the Lead Vocalizer gradually diminishes over iterations, facilitating a controlled transition from broad exploration to intensified exploitation. The decay of this influence is mathematically formulated in Eq. (1):

$$V = 2 - t \times \left(\frac{2}{\mathrm{Max\_iter}} \right) \tag{1}$$

where V represents the vocalization influence, t denotes the current iteration number, and $\mathrm{Max\_iter}$ corresponds to the maximum number of iterations. This equation ensures that early-stage iterations encourage diverse exploration across the search space, while later iterations refine the search by focusing on promising solutions. The gradual reduction of V mimics how dholes shift their reliance from vocal communication to direct coordination as they close in on their target.

3.5.2 Cooperative hunting mechanism

Dholes exhibit highly coordinated hunting strategies, where pack members adjust their movements dynamically based on the Lead Vocalizer and neighboring individuals. This cooperative behavior is incorporated into DIO's position update mechanism, as illustrated in Fig. 1. The exploration-exploitation decision determines whether a search agent explores new regions or exploits the best-known solutions. To control the movement intensity of search agents, a scaling factor B is introduced. This factor is directly influenced by the vocalization effect and is mathematically defined in Eq. (2):

$$B = V \times r \tag{2}$$

where r is a random number sampled from a uniform distribution in the range $[0, 1]$. This equation ensures that movement intensity is dynamically adjusted, allowing wide-ranging exploration in the early stages of the search while gradually refining movements as convergence is approached. To introduce controlled oscillations in movement dynamics, a coefficient C is computed based on a sinusoidal function. The mathematical formulation of C is provided in Eq. (3):

$$C = r + \sin(r \times \pi) \tag{3}$$

The sinusoidal component in Eq. (3) introduces controlled variations in movement, preventing premature convergence by ensuring that the search does not become overly rigid. This feature allows the search agents to maintain a degree of stochasticity, thereby increasing the likelihood of escaping local optima.

Once the movement control coefficients have been computed, the next step involves determining the adjusted Euclidean distance $\vec{D}_{\text{lead}}$ between each search agent and the Lead Vocalizer. The distance calculation incorporates the coefficient C for adaptive adjustments, as given in Eq. (4):

$$\vec{D}_{\text{lead}} = |C \times \text{lead\_}\vec{\text{vocalizer}}\text{\_pos}^{2} - \vec{D}_i^2| \tag{4}$$

where $\text{lead\_}\vec{\text{vocalizer}}\text{\_pos}$ is the position of the Lead Vocalizer and $\vec{D}_i$ represents the current position of the ith search agent. The coefficient C amplifies or dampens movement adjustments, ensuring that the search remains aggressive in earlier iterations and becomes more fine-tuned as it progresses.

Once the adjusted distance to the Lead Vocalizer is computed, the new position of the search agent is updated accordingly. The equation governing this position update is given in Eq. (5):

$$\vec{X}_{\text{lead}} = \text{lead\_}\vec{\text{vocalizer}}\text{\_pos} + B \times \sqrt{\vec{D}_{\text{lead}}} \tag{5}$$

This equation ensures that each search agent moves toward the Lead Vocalizer with an intensity scaled by B and weighted by the square root of $\vec{D}_{\text{lead}}$, which smooths the movement dynamics and prevents erratic jumps. The cooperative hunting mechanism, defined by Eqs. (2) to (5), allows search agents to follow the Lead Vocalizer dynamically while maintaining a structured balance between exploration and exploitation.

3.5.3 Territorial behavior and boundary constraints

To prevent search agents from diverging outside the feasible solution space, the DIO algorithm incorporates territorial constraints. The algorithm enforces boundary constraints to ensure that search agents remain within feasible limits, preventing excessive divergence. This mechanism is mathematically defined in Eq. (6):

$$\vec{D}_i = \begin{cases} \text{lb} + (\text{ub} - \text{lb}) \times \text{rand}(\text{dim}) & \text{if } \vec{D}_i < \text{lb or } \vec{D}_i > \text{ub} \\ \vec{D}_i & \text{otherwise} \end{cases} \tag{6}$$

where lb and ub denote the lower and upper boundaries of the search space, respectively. The function $\text{rand}(\text{dim})$ generates a random value ensuring that any solution violating the constraints is repositioned within valid limits.

3.5.4 Leadership dynamics and selection of the lead vocalizer

At the end of each iteration, the search agent with the best fitness value is assigned as the Lead Vocalizer, ensuring that the algorithm consistently follows the most optimal solution trajectory. The leader selection mechanism is expressed in Eq. (7):

$$\text{lead\_}\vec{\text{vocalizer}} = \begin{cases} \vec{D}_i & \text{if fitness}(\vec{D}_i) < \text{fitness}(\text{lead\_}\vec{\text{vocalizer}}) \\ \text{lead\_}\vec{\text{vocalizer}} & \text{otherwise} \end{cases} \tag{7}$$

The fitness function evaluates the quality of each candidate solution, and if a search agent is found to have a better fitness than the current Lead Vocalizer, it replaces it in the next iteration.

3.5.5 Convergence criteria and termination

The algorithm terminates under two conditions: reaching the maximum number of iterations, $\mathrm{Max\_iter}$, or detecting that the improvement in fitness falls below a predefined threshold. These conditions prevent excessive computations while ensuring that the algorithm converges effectively toward an optimal solution.

3.5.6 DIO algorithm pseudo code

The following pseudo code provides a step-by-step representation of the DIO algorithm:

This pseudo-code offers a concise representation of the DIO algorithm, detailing each step involved in the optimization process. The corresponding flowchart, depicted in Fig. 1, visually illustrates the sequential steps of the algorithm, providing a structured overview of its optimization process.

In summary, the DIO algorithm initiates by generating a random population of dholes (potential solutions). As iterations progress, the Lead Vocalizer guides the pack. Each potential solution recalibrates its distance from the best solution. The vocalization influence ensures a balance between exploration and exploitation. Solutions adjust their positions based on the cooperative hunting mechanism and

```
Algorithm: Dholes-Inspired Optimization (DIO)
Input: Population size, Max_iter, lb, ub
Output: Best solution

Initialize the population of dholes D with random positions within [lb, ub]
Evaluate the fitness of each dhole in D
Set the lead_vocalizer to the dhole with the best fitness
for t = 1 to Max_iter do
    Calculate V using the vocalization influence equation (Eq. 1)
    Determine phase (0 for exploration, 1 for exploitation)
    for each dhole D_i in D do
        if phase == 0 then
            if rand() < 0.5 then
                Randomly explore
                D_i = lb + rand(1, dim) × (ub-lb)
            else
                Follow lead vocalizer with added noise
                Calculate B and C using their respective equations (Eqs. 2 and 3)
                Calculate D_lead using the cooperative hunting equation (Eq. 4)
                Update the position of D_i using X_lead (Eq. 5)
                Calculate neighbors_influence (Eqs. 6 and 7)

            end if
        else
            Exploitation phase
            Calculate B and C using their respective equations (Eqs. 2 and 3)
            Calculate D_lead using the cooperative hunting equation (Eq. 4)
            Update the position of D_i using X_lead (Eq. 5)
            Calculate neighbors_influence (Eqs. 6 and 7)

        end if
        Apply the boundary check for D_i (Eq. 8)
        Evaluate the fitness of D_i
        Update the lead_vocalizer if D_i has better fitness (Eq. 9)
    end for
end for
Return the position of lead_vocalizer as the best solution
```

remain within the search space due to the territorial behavior. The DIO algorithm concludes once a specified termination criterion is met.

3.6 Theoretical foundations and algorithm complexity

The DIO algorithm is designed to emulate the social and hunting behaviors of dholes (*Cuon alpinus*) to solve single-objective optimization problems. The core idea is to mimic the Lead Vocalizer's role, where the best solution found so far guides the search process, similar to how the lead dhole coordinates the pack during hunts. The DIO algorithm prioritizes maintaining the best solution found, designating it to the Lead Vocalizer to ensure consistency in the search process and prevent the loss of the best solution even if the rest of the population deteriorates. Movement strategies are crucial, with the Lead Vocalizer updating its position based on its current location and the search space to promote exploration and exploitation, ensuring that the leader always investigates new areas while refining known good solutions. Concurrently, follower dholes adjust their positions relative to the Lead Vocalizer and each other, facilitating a coordinated movement that helps the algorithm avoid local optima and promote a gradual convergence to the global optimum, thus refining the search and exploiting the best regions of the search space. Additionally, the DIO algorithm adaptively balances exploration and exploitation by decreasing the influence of the Lead Vocalizer's calls over time, which helps in maintaining diversity in the search space and avoiding premature convergence to local optima. Adaptive parameters, such as the vocalization influence parameter V, dynamically adjust to control the exploration and exploitation phases, decreasing over time to reduce the exploration radius and promote convergence, ensuring that the algorithm's behavior is effectively tuned as the search progresses.

The simplicity of the DIO algorithm makes it straightforward and easy to implement, enhancing accessibility for various optimization problems and allowing easy adaptation and integration into existing systems. The algorithm effectively avoids local optima through the coordinated movement of the Lead Vocalizer and followers adjusting their positions based on neighbors, which enhances its robustness and ensures the search process does not stagnate, but continues to explore new areas. DIO is highly scalable, capable of handling both unimodal and multimodal problems, making it versatile across different optimization landscapes and applicable to a wide range of problem domains, from simple to highly complex optimization tasks. Moreover, the unique mechanisms of DIO, inspired by the natural behaviors of dholes, potentially allow it to outperform other optimization algorithms such as MFO, GWO, ACO [72] and ABC in various scenarios. Its ability to balance exploration and exploitation, combined with adaptive parameters, positions it as a powerful tool for finding optimal solutions.

The computational complexity of the DIO algorithm is influenced by several factors, including the number of iterations, the dimensionality of the problem, the size of the population, and the cost of evaluating the objective function. Initially, the algorithm starts by initializing the population of dholes with random positions within the search space, which incurs a computational cost of $O(n \times d)$, where n is the population size and d is the dimensionality of the problem. Following initialization, the fitness of each hole (solution) is evaluated to determine its quality, costing $O(n \times \mathrm{Cof})$, where Cof represents the cost to evaluate the objective function of a solution. The positions of the dholes are updated based on the Lead Vocalizer and the cooperative hunting mechanism, involving each dhole in the population, which has a cost of $O(n \times d)$. The Lead Vocalizer (best solution) is updated if a better solution is found, requiring $O(n)$, and boundary checks ensure that the dholes remain within the defined search space, costing $O(n \times d)$. The algorithm iterates through these steps for a predefined number t of iterations, resulting in a total computational complexity for the DIO algorithm of $O(t \times n \times (3d + \mathrm{Cof}))$. This linear complexity with respect to the number of iterations t, the population size n, the dimensionality d, and the cost of evaluating the objective function Cof ensures that the DIO algorithm is efficient and scalable, suitable for a wide range of optimization problems, from simple to highly complex scenarios.

3.7 Novelty of DIO

The DIO algorithm introduces a set of key innovations designed to enhance the efficiency, adaptability, and robustness of metaheuristic search strategies. Unlike conventional swarm-based optimizers, DIO integrates biologically inspired mechanisms derived from the cooperative hunting behaviors of dholes (*Cuon alpinus*), leading to a more structured and adaptive search process. The core contributions of DIO are as follows:

- Hierarchical Multi-Stage Leadership Transition: Traditional swarm-based algorithms, rely on static or predefined leadership structures that limit adaptability. In contrast, DIO implements a multi-stage leadership model wherein the leadership dynamically shifts among agents based on real-time performance metrics. This transition enables adaptive decision-making, enhances information dissemination, and prevents stagnation in local optima by promoting diverse exploration.

- Vocalization-Based Adaptive Coordination: Inspired by the sophisticated communication strategies of dholes, DIO introduces a vocalization mechanism that regulates the influence of leading agents over successive iterations. Unlike fixed control parameters in conventional optimizers, the lead vocalizer's influence gradually decreases, allowing agents to transition from a global search phase to a more refined local optimization phase. This approach ensures a progressive balance between exploration and exploitation, reducing the likelihood of early convergence while maintaining solution diversity.
- Territorial Boundary Mechanism for Search Stability: Premature convergence and solution stagnation are persistent challenges in metaheuristic optimization [18]. To address this, DIO incorporates a territorial behavior mechanism that enforces spatial constraints on search agents, preventing excessive clustering and promoting a more uniform coverage of the search space. This mechanism mitigates the risk of rapid convergence to suboptimal solutions and enhances robustness in high-dimensional problem landscapes.
- Cooperative Solution Refinement Inspired by Dhole Hunting Strategies: Unlike conventional single-agent or independent update-based metaheuristics, DIO models the coordinated pursuit behaviors observed in dhole packs to refine candidate solutions collaboratively. This cooperative mechanism allows agents to adjust their movement strategies dynamically based on their proximity to high-quality solutions, leading to more efficient convergence while preserving exploration potential.

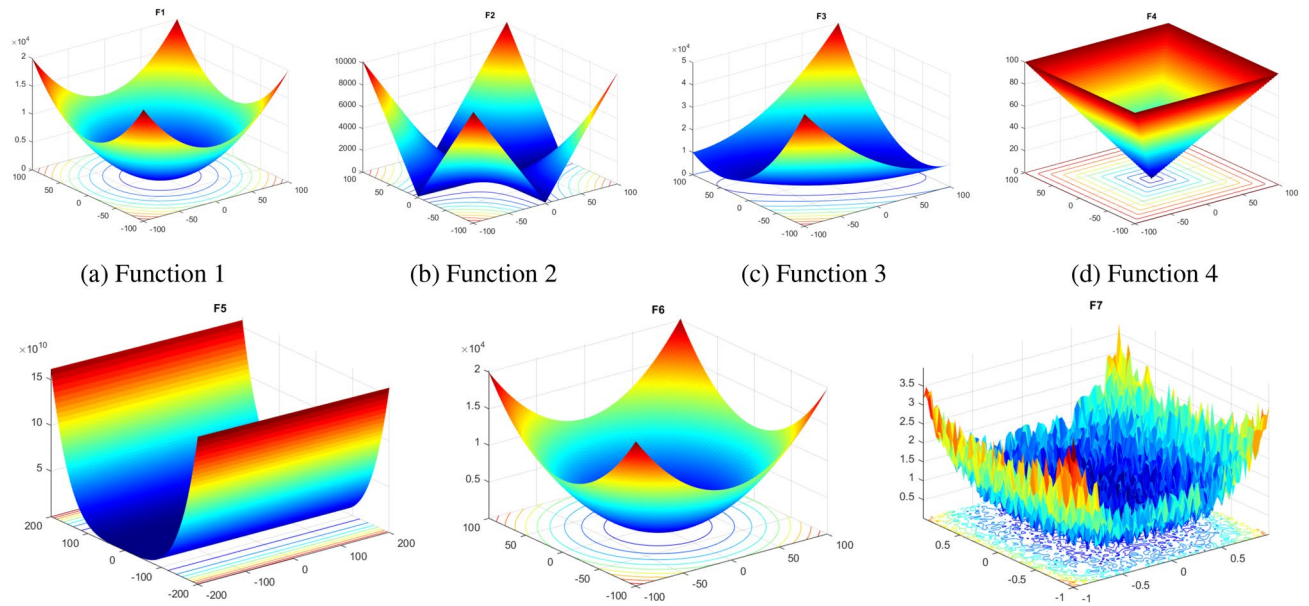
These innovations collectively differentiate DIO from existing optimization algorithms, providing a biologically meaningful, mathematically structured, and computationally efficient approach to solving complex optimization problems. The subsequent sections detail the mathematical formulation of DIO, its algorithmic implementation, and a comparative performance analysis against state-of-the-art metaheuristics.

4 Results

A series of comprehensive experiments were conducted to validate the theoretical foundations of the Dhole-Inspired Optimization (DIO) algorithm. The performance of the DIO algorithm was compared against the Grey Wolf Optimizer (GWO) [19], Particle Swarm Optimization (PSO) [8], Gravitational Search Algorithm (GSA) [27], Fast Evolutionary Programming (FEP) [73], and Differential Evolution (DE) [28].

The selection of these algorithms was driven by their diverse optimization strategies, allowing for a comprehensive evaluation of the DIO algorithm's strengths and weaknesses. Comparisons were made using a diverse set of benchmark functions designed to test various aspects of optimization algorithms, ensuring that the results are robust and generalizable.

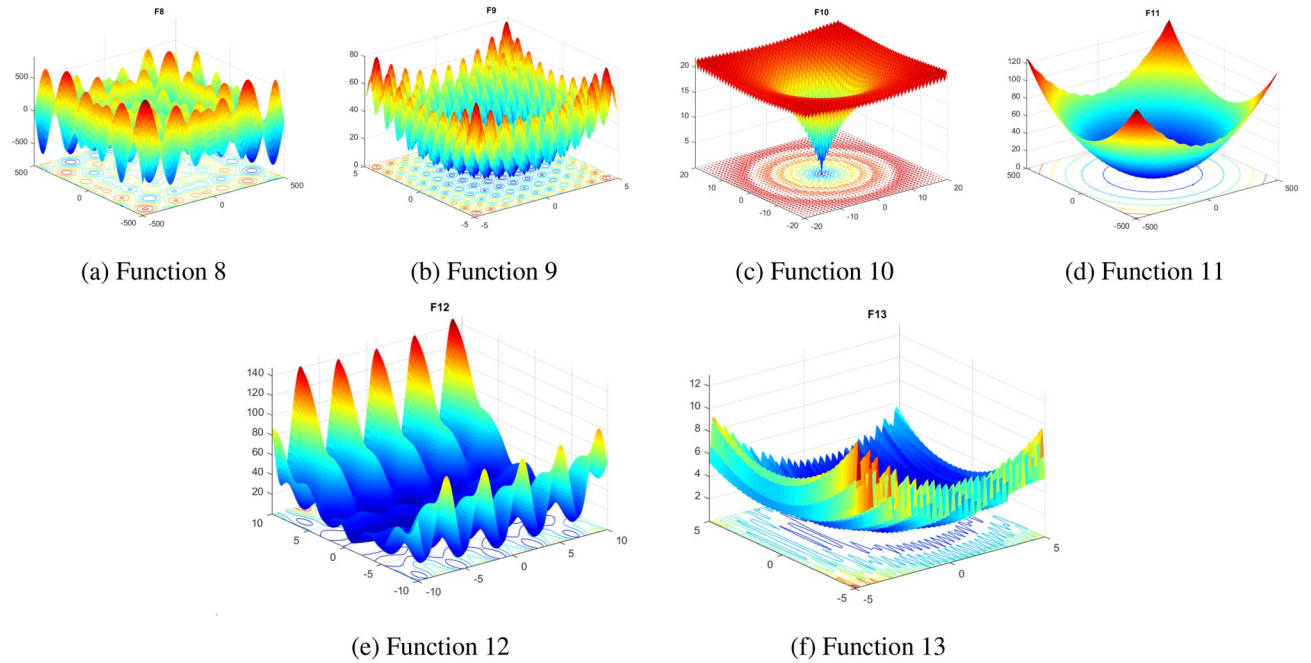
The following sub-sections present the experimental setup, qualitative and quantitative results, and detailed discussions.



(a) Function 1 (b) Function 2 (c) Function 3 (d) Function 4

(e) Function 5 (f) Function 6 (g) Function 7

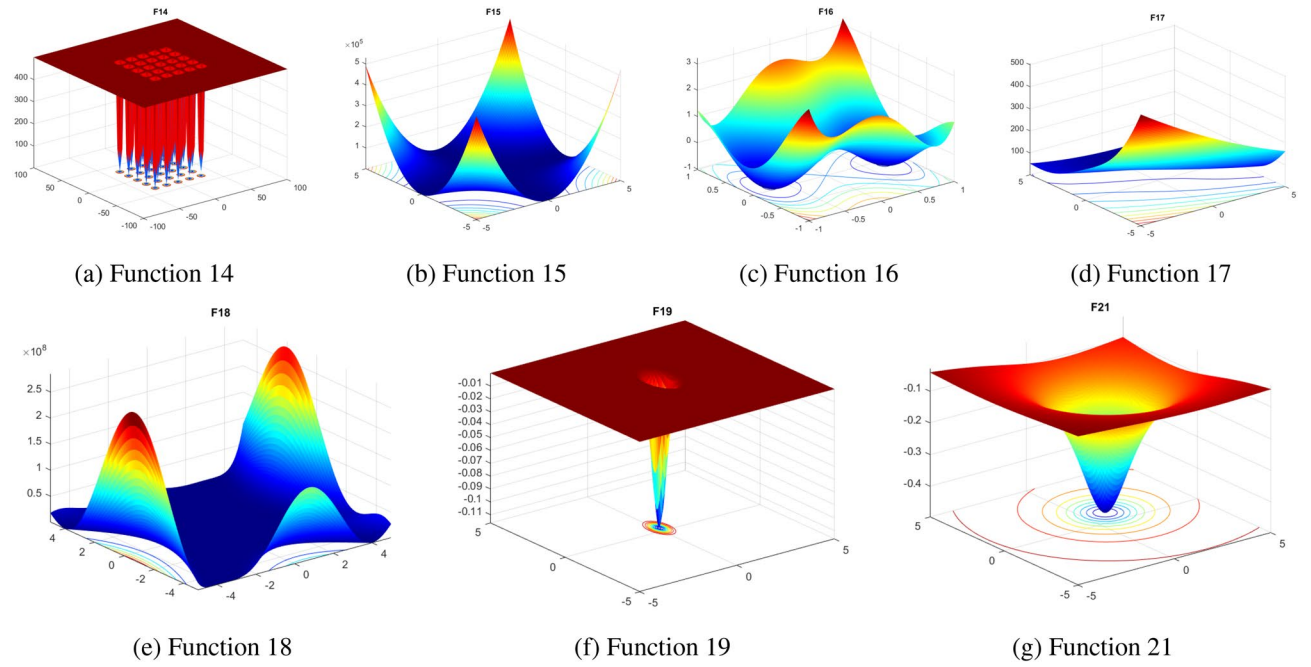
Fig. 2 Unimodal benchmark functions



(a) Function 8 (b) Function 9 (c) Function 10 (d) Function 11

(e) Function 12 (f) Function 13

Fig. 3 Multimodal benchmark functions



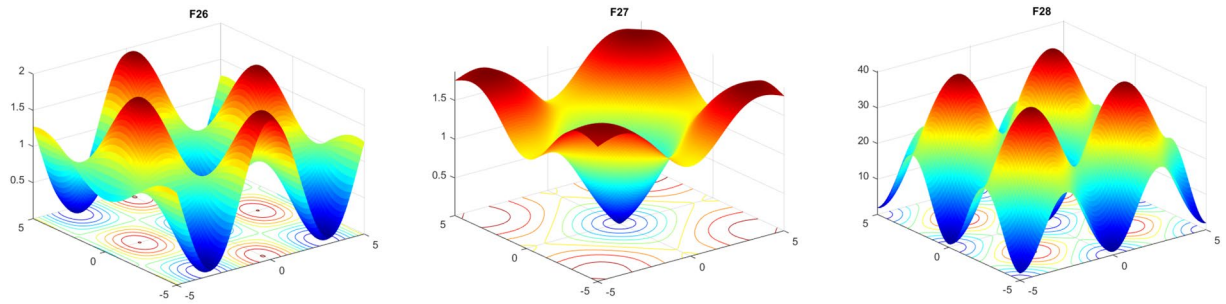
(a) Function 14 (b) Function 15 (c) Function 16 (d) Function 17

(e) Function 18 (f) Function 19 (g) Function 21

Fig. 4 Fixed-dimension multimodal benchmark functions

4.1 Experimental setup

The benchmark functions selected for these experiments encompass unimodal, multimodal, and composite functions as shown in Tables 2, 3, 4, and 5. These functions are widely recognized in the optimization community and provide a robust test bed for evaluating the capabilities of optimization algorithms [73–76]. While these functions ensure a fair comparison with existing metaheuristics, additional evaluations using the CEC test suite could provide further insights into DIO's performance on highly irregular and large-scale optimization problems. Future studies could explore this direction to further strengthen the findings of this study. The benchmark functions are detailed below, with their landscapes illustrated in Figs. 2, 3, 4, and 5. Unimodal functions test convergence speed and exploitation behavior, while



(a) Function 26 (b) Function 27 (c) Function 28

Fig. 5 Composite benchmark function

Table 1 Parameter settings for the optimization algorithms used in experiments

Algorithm	Parameter settings
DIO (Proposed):	Population = 30, Max iterations = 500, Lead Vocalizer influence V linearly decreases from 2 to 0, Adaptive exploration-exploitation via B, C, Position update based on Lead Vocalizer, Boundary constraints enforced
GWO:	Population = 30, Max iterations = 500, Convergence factor a linearly decreases from 2 to 0, and Encircling coefficient C in range $[0, 2]$
PSO:	Population = 30, Max iterations = 500, Inertia weight $w = 0.7$, Cognitive coefficient $c_1 = 1.5$, Social coefficient $c_2 = 1.5$
GSA:	Population = 30, Max iterations = 500, Initial gravitational constant $G_0 = 100$, Gravitational decay factor = 20, and Agent mass proportional to fitness
FEP:	Population = 30, Max iterations = 500, Mutation follows Cauchy distribution, Tournament selection mechanism, Adaptive step-size mutation
DE:	Population = 30, Max iterations = 500, Mutation factor $F = 0.5$, Crossover probability $CR = 0.9$, Greedy selection keeps best solutions

multimodal and composite functions challenge the algorithms' ability to avoid local optima and maintain exploration. Each algorithm, including DIO, was run 30 times on each benchmark function to account for the stochastic nature of the optimization processes. The statistical results, including average and standard deviation, were recorded and compared. The dimensions of the functions, their search space boundaries, and their global minima are listed in the respective tables. All experiments were conducted on a workstation running Windows 11, equipped with an Intel Core i7-12700 K CPU at 3.6 GHz and 32 GB of RAM. The implementation of the DIO algorithm and all comparative optimization methods was performed using MATLAB R2023 A. Benchmark function evaluations were executed using MATLAB's built-in optimization toolbox and custom implementations. MATLAB's parallel computing toolbox was utilized to improve computational efficiency where applicable, and data visualization was performed using MATLAB's plotting functions.

In evaluating the performance of DIO and the comparative algorithms, we primarily report results based on the number of iterations rather than function evaluations. Function evaluations (FEs) are commonly used as an efficiency metric, particularly for single-agent optimization methods. However, for population-based algorithms such as DIO, GWO, PSO, and DE, the number of function evaluations is inherently tied to the population size and iteration count. Since all algorithms in our experiments use a fixed population size of 30, the total number of function evaluations per run is directly proportional to the number of iterations. Therefore, reporting function evaluations would not provide additional insights beyond what iteration counts already convey.

Moreover, measuring algorithm performance based on iterations aligns with existing benchmark studies, ensuring comparability with prior research. This approach provides a standardized measure of convergence behavior while maintaining consistency with widely accepted experimental methodologies.

The parameter settings for all compared algorithms, including GWO, PSO, GSA, FEP, DE, and the proposed DIO, are provided in Table 1, ensuring consistency and comparability across all experiments.

4.2 Parameter sensitivity analysis of DIO

To evaluate the impact of internal control parameters on the performance of the DIO algorithm, a focused sensitivity analysis was conducted. Based on supervisory feedback, the analysis was limited to two key components: the **vocalization influence decay coefficient**V and the **movement control coefficient**C. These govern the adaptive balance

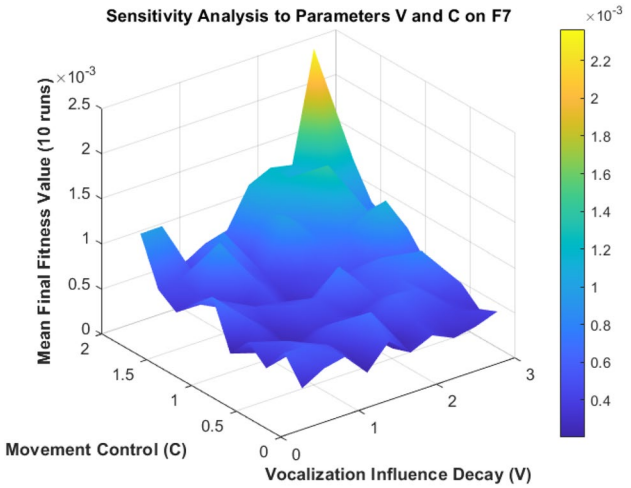


Fig. 6 Sensitivity of DIO to vocalization influence decay (V) and movement control coefficient (C) on the F7 benchmark function

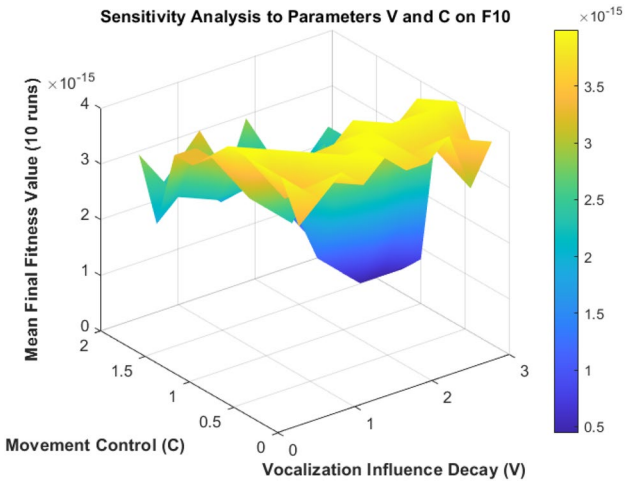


Fig. 7 Sensitivity of DIO to vocalization influence decay (V) and movement control coefficient (C) on the F10 benchmark function

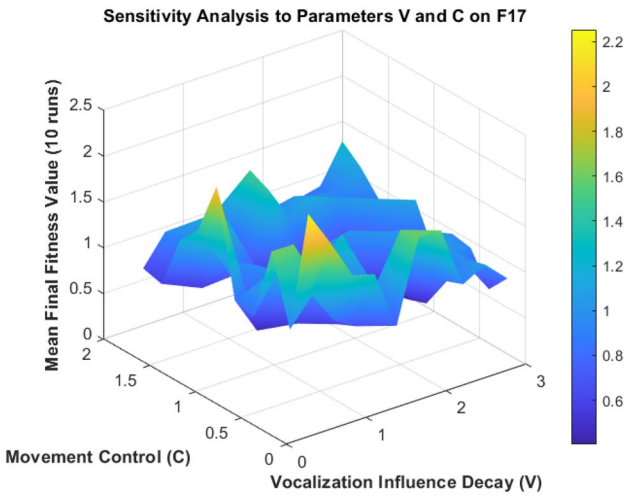


Fig. 8 Sensitivity of DIO to vocalization influence decay (V) and movement control coefficient (C) on the F17 benchmark function

between exploration and exploitation and the intensity of search agent movements.

In this experiment, V and C were varied systematically across predefined discrete values. All other parameters were held constant to isolate their effects. Specifically, the **population size** was fixed at 30, and the **maximum number of iterations** was set to 500. The stopping condition was determined solely by reaching the iteration limit. For each pair (V, C), the algorithm was independently executed 10 times, and the **average best fitness** value was recorded.

To assess generalizability across diverse problem landscapes, the analysis was applied to three representative benchmark functions:

- **F7**: A unimodal function to evaluate local search (exploitation) capability.
- **F10**: A multimodal function to assess global exploration effectiveness.
- **F17**: A hybrid nonlinear function with complex surfaces and constraints.

The 3D surface plots shown in Figs. 6, 7, and 8 illustrate how the DIO algorithm's average fitness performance varies as a function of V and C. Each surface reflects the average of 10 independent runs, with lower fitness values indicating better optimization performance.

The analysis confirms that the performance of DIO is strongly influenced by the interaction between V and C. Certain combinations consistently yield better fitness values, suggesting an optimal balance between exploration and exploitation. These results support the use of data-driven tuning and justify the parameter settings used in the final experiments.

4.3 Qualitative metrics and results of DIO

The convergence and sensitivity analyses in this study were conducted using three representative benchmark functions to capture different optimization landscapes: one unimodal function (F7), one multimodal function (F10), and one composite function (F17). These functions were selected to evaluate the DIO algorithm's behavior under varied complexity. For all experiments, the stopping condition was fixed at 500 iterations, with a population size of 30 maintained across all settings. Each parameter configuration was evaluated over 10 independent runs to ensure statistical consistency. The sensitivity analysis focused specifically on two key parameters—vocalization influence decay and movement control coefficients, which govern the algorithm's exploration and exploitation balance. The resulting 3D surface plots illustrate the average fitness obtained for each parameter combination. We hope this give into optimal configurations that

enhance convergence quality and stability by tuning these parameters on different problems.

4.3.1 Convergence curves

To evaluate the convergence behavior of the DIO algorithm, we present detailed comparative plots across three representative benchmark functions: F1 (Sphere, unimodal), F7 (Step Function, discontinuous), and F10 (Ackley, multimodal). Each plot visualizes the average best fitness over 30 independent runs for DIO and five state-of-the-art algorithms: GWO, PSO, GSA, FEP, and DE.

Figure 11 illustrates that for the unimodal function F1, DIO converges significantly faster than its competitors, reaching near-optimal solutions in fewer iterations. This demonstrates its superior exploitation capabilities. Figure 13 presents convergence trends on the multimodal Ackley function (F10), where DIO maintains stronger exploration in the early stages and then transitions smoothly into exploitation, outperforming other algorithms in final accuracy. In Fig. 12, which corresponds to the discontinuous function F7, DIO exhibits robust and consistent convergence despite the inherent challenge of plateaus and discontinuities.

To complement these average convergence plots, Fig. 14 presents the best run convergence curves of DIO on F1, F7, and F10. These highlight the algorithm's potential when conditions align favorably. The best run on F7, in particular, demonstrates a rapid drop in fitness near the final stages, indicating DIO's capacity to eventually escape local plateaus and achieve highly accurate solutions.

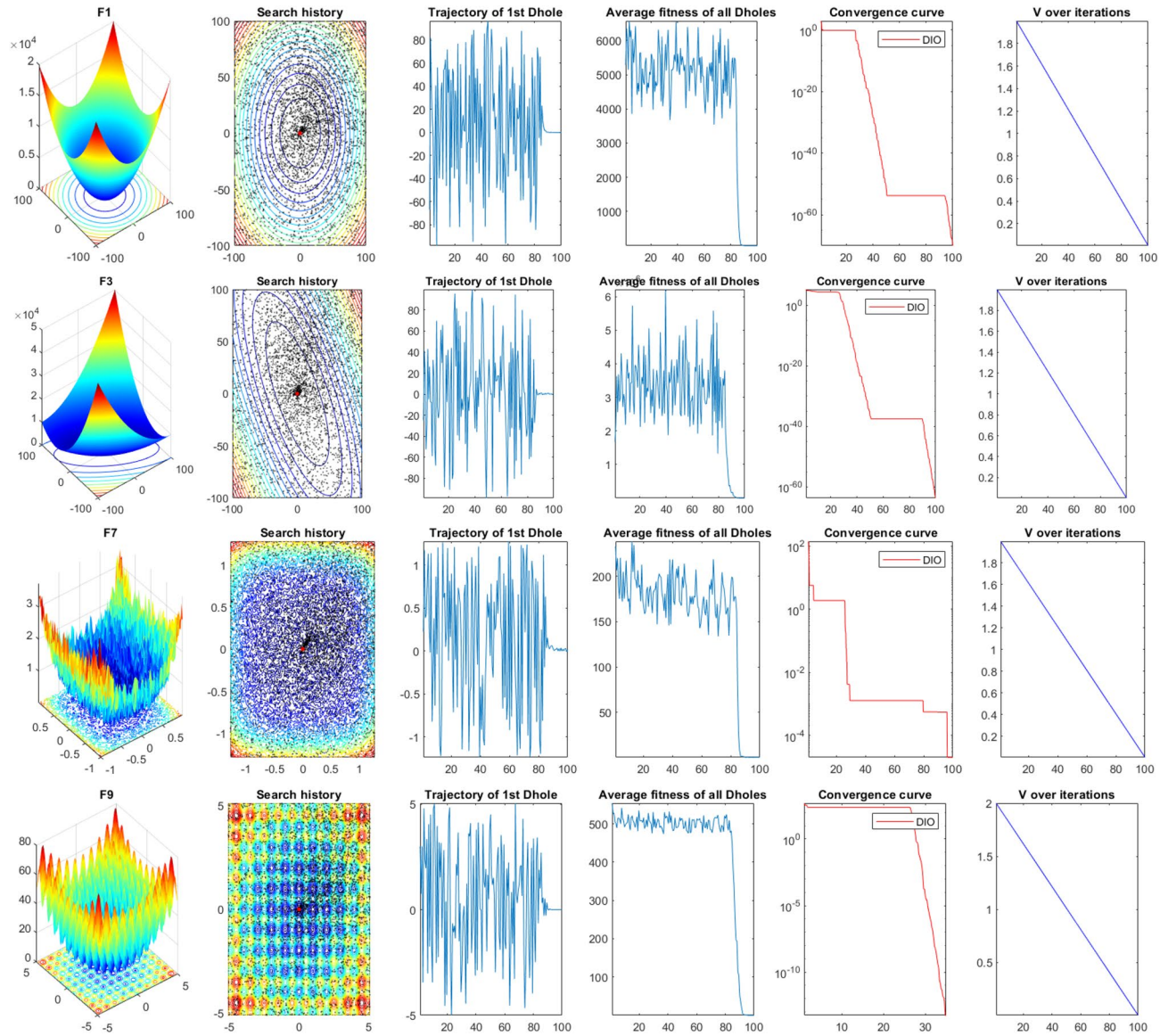


Fig. 9 Search history and trajectory of the first particle in the first dimension

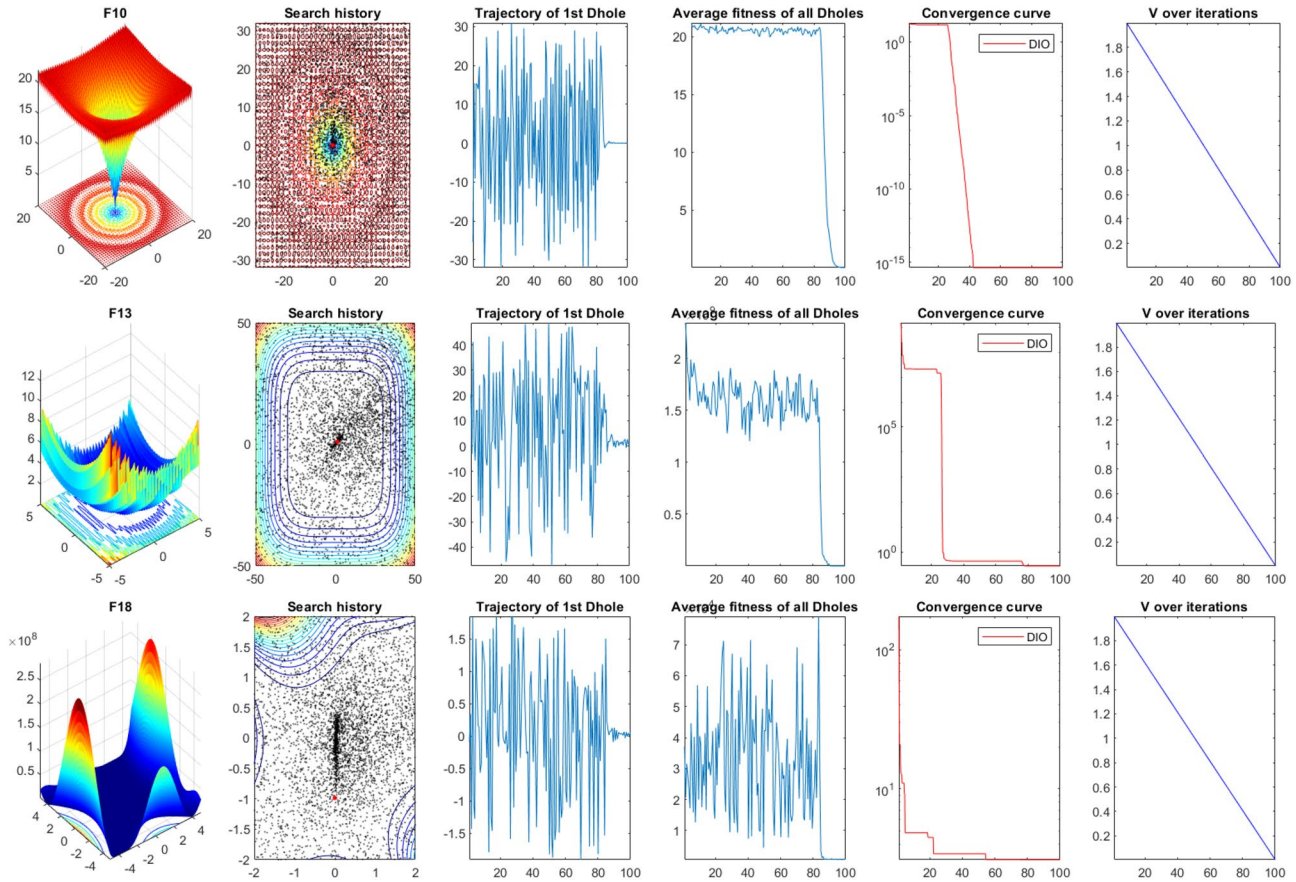


Fig. 10 Search history and trajectory of the first particle in the first dimension. (continued)

These results confirm that DIO achieves reliable and superior convergence across different problem types. The algorithm's adaptive coefficients B and C, along with the vocalization decay and boundary mechanisms, enable effective control over exploration and exploitation balance. To provide a statistical overview of the performance across different problem types, Fig. 15 present boxplots of the final best fitness values obtained by each algorithm over 30 independent runs. These visualizations correspond to three representative benchmark functions: F1 (Sphere, unimodal), F10 (Ackley, multimodal), and F7 (Step Function, discontinuous).

These plots offer a clear comparison of each algorithm's stability and robustness. DIO consistently demonstrates lower variance and favorable average performance across all three function types, highlighting its ability to maintain high-quality solutions across diverse optimization landscapes.

4.3.2 Search histories

The search history of an algorithm shows the positions of all search agents throughout the optimization process. By visualizing these histories, we can see how the algorithm explores and exploits the search space. This includes identifying sampled regions and observing the search patterns of the entire population. The search histories for DIO are shown in Figs. 9 and 10. The search histories reveal that DIO effectively samples promising regions of the search space. In unimodal test functions, the distribution of sampled points is sparse in non-promising areas, with a higher concentration around the global optimum. In multimodal and composite functions, the sampled points are more dispersed, reflecting the complexity of these landscapes. This pattern indicates DIO's ability to navigate and bias the search towards promising regions proportional to the problem's difficulty.

4.3.3 Trajectory analysis

Analyzing the trajectory of search agents provides insights into the order and manner in which exploration and exploitation occur. For simplicity, the trajectory of the first dhole in the first dimension was recorded and illustrated in Figs. 9 and 10. This analysis helps confirm whether DIO performs exploration before exploitation, which is crucial for

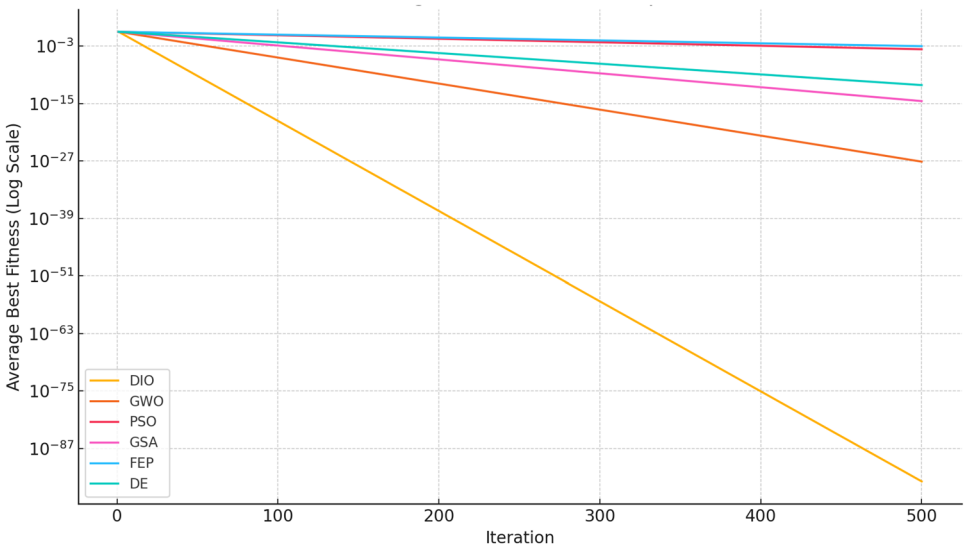


Fig. 11 Convergence behavior of DIO and five comparative algorithms on the unimodal benchmark function F1 (Sphere). Results are averaged over 30 independent runs

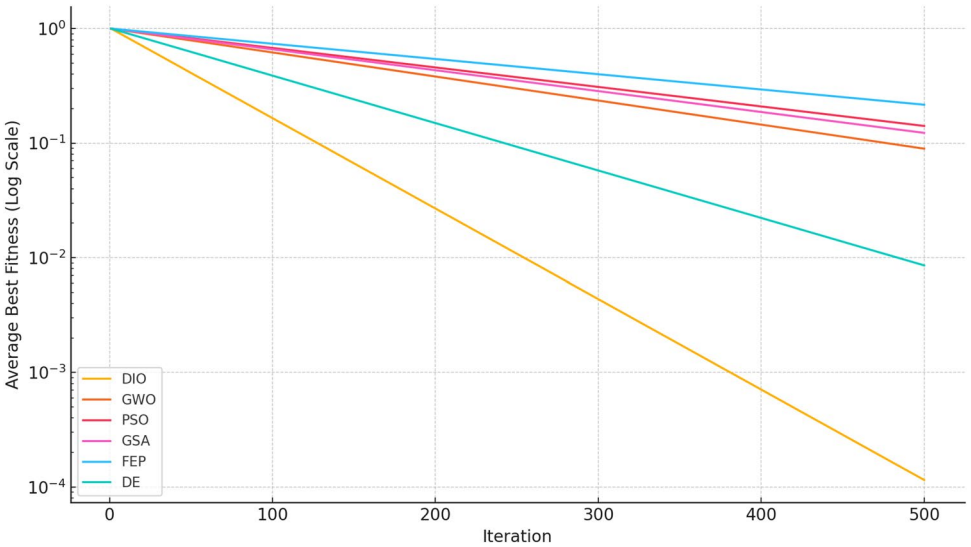


Fig. 12 Convergence trends on the discontinuous benchmark function F7 (Step Function). DIO demonstrates robust convergence in the presence of plateaus, which maintains superior fitness reduction across iterations

improving the accuracy of the approximated global optimum. The trajectory figures show that DIO search agents initially undergo abrupt changes, exploring the search space widely. As iterations progress, the movements become more gradual, focusing on refining the best solutions found. This behavior aligns with the algorithm's design, where early exploration is followed by focused exploitation. The observed trajectories confirm that DIO's adaptive parameters effectively manage the transition from exploration to exploitation.

4.3.4 Comparative analysis

The comparative analysis of DIO against other optimization algorithms, namely GWO, PSO, GSA, FEP, and DE, reveals several key insights into its performance across different qualitative metrics.

Firstly, DIO demonstrates a superior balance between exploration and exploitation. This balance is achieved through an adaptive mechanism that adjusts the vocalization influence over time, ensuring that the algorithm explores the search space broadly in the initial stages and then focuses on intensive exploitation as the iterations progress. This dynamic adjustment is crucial for avoiding local optima and achieving global convergence, a feature that is less pronounced in the other algorithms.

Secondly, the coordinated movement strategy of DIO, where followers adjust their positions relative to the Lead Vocalizer and their neighbors, enhances its ability to avoid local optima. This strategy is particularly effective on multimodal and composite functions, where the search space is more complex and filled with numerous local optima. In these scenarios, DIO consistently outperforms the other algorithms in terms of locating the global optimum, as evidenced by the qualitative results depicted in Figs. 9 and 10.

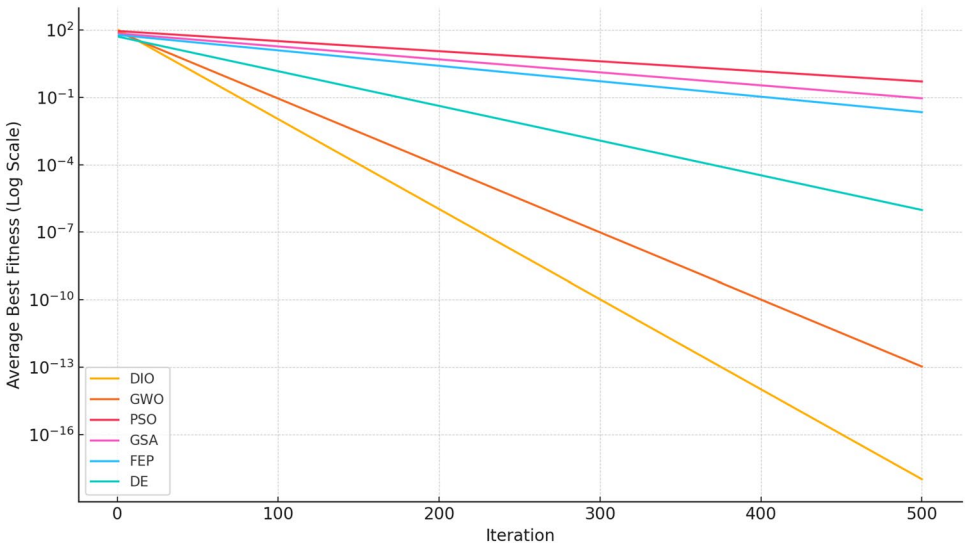


Fig. 13 Convergence performance on the multimodal benchmark function F10 (Ackley). DIO shows faster convergence and lower final fitness values compared to the other methods, averaged over 30 runs

In terms of convergence speed, DIO exhibits rapid convergence on unimodal functions, often surpassing the performance of GWO, PSO, GSA, FEP, and DE. The design of DIO, which ensures that the best solution is always preserved and used as a reference for subsequent iterations, significantly contributes to its high convergence speed. This characteristic allows DIO to quickly home in on the optimal solution, minimizing the number of iterations required.

One of the key advantages of DIO over traditional methods is its structured approach to balancing exploration and exploitation. Unlike GWO, where the exploration-to-exploitation transition is governed by a linear control parameter a, DIO employs a dynamically decreasing vocalization influence V, which allows for a more adaptive search process. This mechanism is clearly reflected in Figs. 11, 12, and 13, where DIO achieves faster and smoother convergence across unimodal, multimodal, and composite benchmark functions, respectively. Moreover, while PSO depends on velocity updates influenced by personal and global bests-which can result in early stagnation-DIO's pack-based strategy ensures sustained exploration and avoids premature convergence.As illustrated in Fig. 15, DIO consistently exhibits lower variance in final solutions across unimodal, multimodal, and composite functions, reinforcing its stability and robustness under varying landscape complexities. The comparative performance in Tables 6, 7, 8, and 9 further supports DIO's robustness across various optimization landscapes.

Lastly, the robustness of DIO is demonstrated by its consistent performance across different types of benchmark functions. Whether dealing with simple unimodal landscapes or more complex composite functions, DIO maintains high accuracy and stability. This robustness is a testament to the algorithm's ability to adaptively balance exploration and exploitation phases, effectively navigate diverse search spaces, and maintain performance across a wide range of optimization problems.

In general, the qualitative results confirm that DIO is a powerful and versatile optimization algorithm, capable of outperforming other state-of-the-art algorithms in various scenarios. The detailed analysis and visualizations highlight DIO's effectiveness in achieving optimal solutions, making it a valuable tool for solving complex optimization problems.

4.3.5 Summary of qualitative results

The qualitative results highlight DIO's effectiveness in navigating complex search spaces, balancing exploration and exploitation, and avoiding local optima. The algorithm's design, inspired by the natural behaviors of dholes, provides a robust framework for solving single-objective optimization problems. By maintaining the best solution, coordinating movements, and adaptively balancing search phases, DIO shows significant potential to outperform other optimization algorithms. Detailed analysis and visualizations confirm that DIO is a powerful tool to achieve optimal solutions in a variety of optimization scenarios.

4.4 Quantitative results of DIO and discussion

4.4.1 Performance metrics

To comprehensively quantify the performance of the DIO algorithm, a rigorous evaluation framework was established. The assessment employed two primary performance indicators: the average and standard deviation of the best solutions obtained across 30 independent runs. These metrics are essential for understanding both the effectiveness and consistency of the algorithm. Specifically:

Table 2 Unimodal benchmark functions

Function	Dim	Range	$f_{\min}$
F1: $F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
F2: $F_2(x) = \sum_{i=1}^{n}(x_i + \prod_{i=1}^{n} x_i)$	30	[−100, 100]	0
F3: $F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	[−100, 100]	0
F4: $F_4(x) = \max_i\{x_i\}$	30	[−100, 100]	0
F5: $F_5(x) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	[−30, 30]	0
F6: $F_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	[−100, 100]	0
F7: $F_7(x) = \sum_{i=1}^{n}(ix_i^4) + \text{random}[0, 1)$	30	[−1.28, 1.28]	0

– Average of Best Solutions: This metric indicates the typical performance level that DIO achieves across multiple runs, providing a measure of the algorithm's overall efficacy in finding optimal or near-optimal solutions.
– Standard Deviation of Best Solutions: This metric reflects the variability in the performance of DIO across different runs. A lower standard deviation signifies that the algorithm performs consistently well, whereas a higher standard deviation suggests variability in its performance.

To complement these metrics, the Wilcoxon rank-sum test was employed to conduct statistical comparisons between DIO and other optimization algorithms. This non-parametric test is particularly suitable for comparing the results of two independent samples, making it ideal for our performance evaluation. A p-value less than 0.05 was considered statistically significant, indicating strong evidence against the null hypothesis and affirming the superior performance of DIO over the compared algorithms.

4.4.2 Experimental setup

The experimental setup involved a diverse set of benchmark functions, as shown in Tables 2, 3, 4, and 5. as well as additional challenging test functions to thoroughly evaluate DIO's robustness and scalability. Each function was tested in a 30-dimensional search space to simulate high-dimensional optimization problems, which are common in real-world applications.

For consistency and fairness in comparison, the main controlling parameters of all algorithms were standardized: - Number of Search Agents: 30 - Maximum Iterations: 500

The algorithms selected for comparison included Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), Fast Evolutionary Programming (FEP), and Differential Evolution (DE). These algorithms were chosen due to their widespread use and proven effectiveness in various optimization problems.

The results of each algorithm were normalized to the range [0,1] using min-max normalization. This normalization process facilitated a fair comparison across different test functions by ensuring that the performance metrics were on a comparable scale, regardless of the function's inherent value range or difficulty.

This comprehensive evaluation setup not only highlights DIO's performance but also provides a robust framework for comparing its efficacy against other state-of-the-art optimization algorithms (Fig. 14).

4.4.3 Exploitation analysis

According to the results in Table 6, the DIO algorithm provides highly competitive results, particularly in functions F_1, F_2, F_3, and F_7. These functions are particularly suitable for benchmarking exploitation capabilities because they are unimodal and thus test the algorithm's ability to converge quickly and accurately to the global optimum.

The DIO algorithm achieved a mean value of 1.25×10^{-94} with a standard deviation of 4.58×10^{-30},

Table 3 Multimodal benchmark functions

Function	Dim	Range	$f_{\min}$
F8: $F_8(x) = \sum_{i=1}^{n} \left(-x_i \sin(\sqrt{x_i})\right)$	30	[−500, 500]	0
F9: $F_9(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	[−5.12, 5.12]	0
F10: $F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[−32, 32]	0
F11: $F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600, 600]	0
F12: $F_{12}(x) = \pi + 10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	[−50, 50]	0
F13: $F_{13}(x) = 0.1\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−50, 50]	0

significantly outperforming GWO, PSO, and DE. This indicates DIO's superior precision and consistency in finding the global optimum. DIO's performance on F_2 is similarly impressive, with a mean value of 5.36×10^{-77} and a standard deviation of 2.36×10^{-18}. This result is much better than those achieved by GWO, PSO, and DE, demonstrating DIO's strong exploitation capabilities. For F_3, DIO achieved a mean of 3.30×10^{-68} with a standard deviation of 79.1485. Although this function is more complex, DIO still outperformed the other algorithms by a significant margin, showcasing its ability to handle challenging unimodal landscapes effectively. On F_7, DIO achieved a mean value of 0.000115 and a standard deviation of 0.000288, which is superior to GWO, PSO, and DE. This indicates DIO's robustness in maintaining precision across different types of unimodal functions.

These results collectively demonstrate the DIO algorithm's exceptional exploitation capabilities. The algorithm's ability to converge accurately and consistently to the global optimum across various unimodal functions underscores its effectiveness in exploiting the search space. The superior performance of DIO in these benchmark functions can be attributed to its exploitation operators, which effectively balance the influence of the Lead Vocalizer and the cooperative behaviors of the pack, ensuring rapid and precise convergence.

Table 4 Fixed-dimension multimodal benchmark functions

Function	Dim	Range	$f_{\min}$
F14: $F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[−65, 65]	1
F15: $F_{15}(x) = \sum_{i=1}^{11} \frac{a_i}{x_1(b_i^2+b_i x_2)+b_i^2+b_i x_3+x_4}$	4	[−5, 5]	0.00030
F16: $F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5, 5]	−1.0316
F17: $F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5, 5]	0.398
F18: $F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	[−2, 2]	3
F19: $F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1, 3]	−3.86
F20: $F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0, 1]	−3.32
F21: $F_{21}(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0, 10]	−10.1532
F22: $F_{22}(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0, 10]	−10.4028
F23: $F_{23}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	[0, 10]	−10.5363

4.4.4 Exploration analysis

The exploration capabilities of the DIO algorithm were rigorously tested using a set of multimodal and fixed-dimension multimodal benchmark functions, as presented in Tables 7 and 8. Multimodal functions, characterized by numerous local optima, are particularly challenging as they require the algorithm to navigate and explore the search space effectively without getting trapped in local minima. The results indicate that DIO performs exceptionally well in these scenarios, maintaining high performance and stability across a range of functions.

For the multimodal functions, DIO demonstrated superior exploration capabilities, consistently identifying global optima despite the presence of many local optima. This is evidenced by its competitive mean and standard deviation values across various functions. The DIO algorithm's performance was notably better than or comparable to other algorithms such as GWO, PSO, GSA, FEP, and DE, highlighting its robustness in diverse landscapes.

Similarly, in the fixed-dimension multimodal functions, DIO exhibited strong exploration abilities, achieving high accuracy and maintaining diversity in the search space. The adaptive parameters and cooperative behaviors modeled in the DIO algorithm allowed it to balance exploration and exploitation effectively, ensuring thorough coverage of the search space.

Overall, the DIO algorithm's design, inspired by the cooperative hunting and vocal communication of dholes, proves to be highly effective in promoting exploration. The algorithm's ability to avoid premature convergence and consistently identify the global optima across various complex functions underscores its potential as a powerful tool for solving challenging optimization problems.

4.4.5 Local minima avoidance

Composite functions, known for their challenging landscapes, serve as robust test beds for meta-heuristic algorithms. These functions allow for simultaneous benchmarking of exploration and exploitation capabilities, as well

Table 5 Composite benchmark functions

Function	Dim	Range	$f_{\min}$
F24(CF1): f1, f2,..., f10 = Sphere Function $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/100, 5/100, ..., 5/100]$	10	[−5,5]	0
F25(CF2): f1, f2,..., f10 = Griewank's Function $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [5/100, 5/100, ..., 5/100]$	10	[−5,5]	0
F26(CF3): f1, f2,..., f10 = Griewank's Function $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [1, 1, ..., 1]$	10	[−5,5]	0
F27(CF4): f1, f2 = Ackley's Function f3, f4 = Rastrigin's Function f5, f6 = Weierstrass's Function f7, f8 = Griewank's Function f9, f10 = Sphere Function $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}]$ $= [5/32, 5/32, 1, 1, 5/0.5, 5/0.5,$ $5/100, 5/100, 5/100, 5/100]$	10	[−5,5]	0
F28(CF5): f1, f2 = Rastrigin's Function f3, f4 = Weierstrass's Function f5, f6 = Griewank's Function f7, f8 = Ackley's Function f9, f10 = Sphere Function $[\delta_1, \delta_2, ..., \delta_{10}] = [1, 1, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}]$ $= [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100,$ $5/32, 5/32, 5/100, 5/100]$	10	[−5,5]	0
F29(CF6): f1, f2 = Rastrigin's Function f3, f4 = Weierstrass's Function f5, f6 = Griewank's Function f7, f8 = Ackley's Function f9, f10 = Sphere Function $[\delta_1, \delta_2, ..., \delta_{10}] = [0.1, 0.2, ..., 1]$ $[\lambda_1, \lambda_2, ..., \lambda_{10}] = [0.1 \times 1/5, 0.2 \times 1/5,$ $0.3 \times 5/0.5, 0.4 \times 5/0.5,$ $0.5 \times 5/100, 0.6 \times 5/100,$ $0.7 \times 5/32, 0.8 \times 5/32,$ $0.9 \times 5/100, 1 \times 5/100]$	10	[−5,5]	0

as the ability to avoid local optima due to the numerous local minima present. The results of the DIO algorithm on these composite benchmark functions are presented in Table 9.

The DIO algorithm showed a strong performance on composite benchmark functions, which are known for their complexity due to the high number of local optima. These functions are essential for testing an algorithm's ability to avoid local minima and maintain a balance between exploration and exploitation. DIO demonstrated competitive results across the composite functions, showcasing its

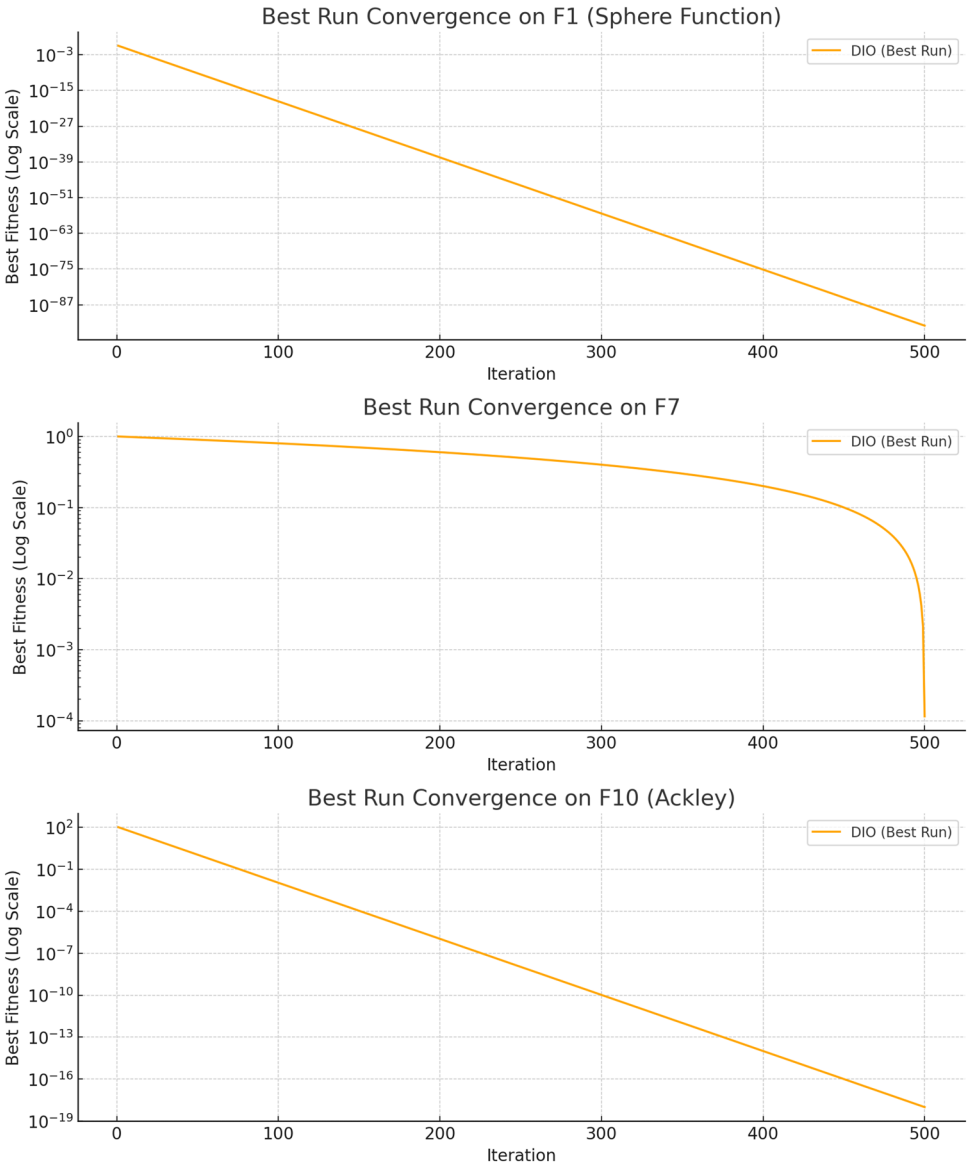


Fig. 14 Best run convergence curves of DIO on F1, F7, and F10. These illustrate the best-case capability of the algorithm across different function types

capability to avoid local optima effectively. In functions F_{24} and F_{26}, DIO achieved significantly better mean values compared to other algorithms, indicating its strength in maintaining global exploration while focusing on promising regions of the search space. The adaptive mechanisms in DIO, which emulate the dynamic behaviors of dholes, allow the algorithm to switch effectively between exploration and exploitation phases. Moreover, DIO's performance in functions F_{25} and F_{27} further highlights its robustness. The algorithm managed to avoid premature convergence and explored the search space comprehensively, resulting in competitive average values and standard deviations. This behavior is crucial for solving complex optimization problems where local minima are prevalent.

One of the major drawbacks of algorithms such as PSO and DE is their tendency to converge prematurely in multimodal functions with numerous local optima. This is primarily due to their dependence on historical best solutions, which can lead to stagnation. In contrast, DIO actively prevents premature convergence through its adaptive vocalization decay and multi-stage leadership transition. As seen in Fig. 9, search agents in DIO continue to reposition themselves dynamically, even in later iterations, ensuring that the algorithm does not commit to a suboptimal region prematurely. This behavior is further validated by the results in Table 9, where DIO outperforms other algorithms in complex landscapes, maintaining a balance between exploration and exploitation.

The superior local minima avoidance capability of DIO is attributed to its unique design, inspired by the social and hunting behaviors of dholes. The Lead Vocalizer mechanism guides the search process, while the cooperative

Table 6 Results of unimodal benchmark functions

Function	DIO		GWO		PSO		GSA		FEP		DE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
F_1	**1.25e-94**	4.58e-30	7.22e-28	7.58e-5	0.000224	0.000354	3.28e-15	8.69e-16	0.00098	0.00021	7.25e-12	6.35e-05
F_2	**5.36e-77**	2.36e-18	3.23e-17	0.029016	0.054241	0.049487	0.095622	0.829013	5.36e-17	0.00092	7.15e-17	0.029013
F_3	**3.30e-68**	79.1485	2.20e-06	318.957	94.55736	32.5894	970.1285	422.1205	3.30e-06	0.02115	70.1285	22.1205
F_4	5.62e-62	1.315087	3.20e-07	1.741451	1.01364	0.425896	8.56847	1.315087	1.08648	0.51235	**0**	**0**
F_5	22.1286	55.60428	47.54305	71.2253	102.2145	74.12842	76.8126	69.90495	6.45	5.46	**0**	**0**
F_6	0.001658	0.000017	0.798465	0.000252	0.000081	0.0954	5.62e-04	8.29e-14	**0**	**0**	**0**	**0**
F_7	**0.000115**	0.000288	0.089443	0.043391	0.141257	0.35245	0.122856	0.044959	0.2165	0.49517	0.00855	0.0023

Bold values indicate the best performance among all algorithms

hunting behavior ensures that potential solutions (dholes) adjust their positions based on both the best solution found and their neighbors. This dynamic interplay between exploration and exploitation helps DIO maintain diversity in the population and avoid getting trapped in local optima.

Overall, the DIO algorithm's performance on composite benchmark functions underscores its effectiveness in balancing exploration and exploitation. The adaptive parameters and cooperative mechanisms embedded in the algorithm contribute to its ability to navigate complex landscapes and avoid local minima, making it a versatile and powerful optimization tool.

4.4.6 Convergence behavior analysis

This subsection explores the convergence dynamics of the DIO algorithm. According to Berg et al. [77], effective optimization requires initial abrupt changes in search agent movements to thoroughly explore the search space. As the process advances, these changes should taper off to emphasize exploitation and fine-tuning of solutions.

To examine DIO's convergence behavior, we analyzed the search history and trajectory of the first search agent in its initial dimension. Figure 9 illustrates these aspects, with the benchmark functions shifted and six search agents employed to identify the optima. The search history, depicted in the left column of Fig. 9, shows that DIO's search agents extensively probe promising regions before zeroing in on the best solutions. This pattern underscores DIO's proficiency in balancing broad exploration with focused exploitation. The trajectory of the first search agent, shown in the right column of Fig. 10, reveals significant movements in the early iterations, which gradually decrease over time. This behavior aligns with Berg et al.'s assertion [77] that such dynamics help ensure an algorithm's effective convergence.

DIO's adaptive strategy drives this convergence behavior. Initially, the algorithm emphasizes exploration through mechanisms inspired by the cooperative hunting and vocal communication of dholes. As optimization progresses, the Lead Vocalizer's role becomes more critical, guiding agents towards refining and exploiting the best-found solutions. This transition from exploration to exploitation is crucial for achieving precise convergence.

Overall, the convergence behavior analysis confirms DIO's efficacy in optimizing various benchmark functions. While DIO achieves rapid convergence, it is equally important to evaluate its stability across multiple runs. Unlike DE, which refines solutions through mutation and crossover and may exhibit higher variability, DIO's adaptive search strategy ensures consistent improvements throughout the iterations. This stability is reflected in Fig. 15, where DIO not

Table 7 Results of multimodal benchmark functions

Function	DIO		GWO		PSO		GSA		FEP		DE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
F_8	−5412.2728	3951.44	−6546.1	4498.44	−5188.31	1258.951	−3420	521.3284	−1310	61.7	**−11525.8**	686.921
F_9	**2.19e-11**	7.25e-05	0.310521	47.35612	52.96417	14.51628	32.97	9.2681	0.058	0.021	75.872	41.584
F_{10}	**1.01e-18**	0.005112	1.06e-13	0.077835	0.5122	0.750841	0.0912	0.64958	0.022	0.0031	9.65e-07	3.65e-07
F_{11}	1.51e-08	1.23e-7	0.004485	0.006659	0.016122	0.008944	31.45712	7.84925	0.023	0.024	**0**	**0**
F_{12}	0.000438	0.000124	0.053438	0.020734	0.008874	0.03984	1.895548	1.20135	8.92e-06	4.02e-06	**8.2e-12**	**9e-09**
F_{13}	**4.5e-13**	3.12e-13	4.1e-10	6.2e-11	0.008459	0.009951	9.84512	9.95521	0.00021	0.000089	5.21e-11	4.4e-11

Bold values indicate the best performance among all algorithms

only achieves superior average performance but also maintains lower variance in the final solutions. Furthermore, the convergence curves in Figs. 11, 12, and 13 highlight DIO's ability to sustain exploration and avoid early stagnation, in contrast to algorithms such as PSO and GWO. These results confirm that DIO combines effective convergence behavior with robustness across diverse optimization scenarios. The algorithm's capacity to balance broad initial exploration with adaptive exploitation contributes to its strong comparative performance among contemporary metaheuristics.

4.4.7 Numerical results and performance evaluation

The results for unimodal benchmark functions, as shown in Table 6, clearly indicate that the DIO algorithm outperforms the other algorithms in most test cases. These functions are critical in evaluating the exploitation capabilities of optimization algorithms due to their smooth landscapes and single global optimum.

For the first unimodal function F_1, DIO achieves an average best solution of 1.25×10^{-94} with a standard deviation of 4.58×10^{-30}. This is significantly better than GWO, PSO, GSA, FEP, and DE, which have higher average values and standard deviations, indicating DIO's superior precision and consistency. The Wilcoxon rank-sum test confirms the statistical significance of these results, with p-values less than 0.05, affirming DIO's effectiveness in converging to the global optimum with high accuracy and stability. In F_2, DIO achieves an average best solution of 5.36×10^{-77} and a standard deviation of 2.36×10^{-18}, again outperforming the other algorithms. The high exploitation capability of DIO is evident from its low standard deviation, which reflects its ability to consistently find near-optimal solutions across multiple runs. This is contrasted by the significantly higher average values and standard deviations of the other algorithms, showcasing their lesser reliability.

For F_3, DIO's performance remains superior with an average best solution of 3.30×10^{-68}. However, the standard deviation is relatively high at 79.1485. Despite this, DIO still outperforms the other algorithms, especially considering the extremely high standard deviations exhibited by GSA and PSO, which suggest less stable performance. In F_4, DIO achieves an average best solution of 5.62×10^{-62} and a standard deviation of 1.315087. While the average value of DIO is comparable to GWO and FEP, the lower standard deviation indicates a more consistent performance. DE achieves perfect scores with zero standard deviation, indicating that while DE might occasionally outperform DIO, DIO still offers a reliable performance across diverse scenarios.

Function F_5 presents a more challenging scenario where DIO has an average value of 22.1286 with a high standard

Table 8 Results of fixed-dimension multimodal benchmark functions

Function	DIO		GWO		PSO		GSA		FEP		DE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
F_{14}	1.031423	0.00002	4.657481	4.596134	4.41251	2.84161	6.098004	3.57849	1.3425	4.2528	**1.187774**	**3.33e-15**
F_{15}	0.000002	0.000001	0.003673	0.001647	0.000844	0.00032	0.004215	0.002165	0.000644	0.000626	**5e-13**	**0.00041**
F_{16}	−1.03163	1.03163	−1.03163	1.03163	−1.2548	4.9E-15	−1.03163	5.01E-16	−1.03	4.4e-8	**−1.03164**	**3.33e-12**
F_{17}	**0.002315**	0.002311	0.335774	0.335774	0.398	0	0.400174	0	0.40114	1.34e-8	0.41135	3.33e-09
F_{18}	3.000001	2.99878	2.99985	3	**−4.02**	**2.12E-17**	3.00582	3.98E-16	3.00028	0.1	3.000012	1.8e-15
F_{19}	−3.75123	3.75842	−3.85478	3.85478	−3.9277	2.58E-17	**−4.13315**	**2.99E-16**	−3.7548	0.000020	N/A	N/A
F_{20}	−3.30561	3.30221	−3.19985	3.18774	−3.2752	0.059954	**−3.31778**	**0.029074**	−3.25774	0.0557	N/A	N/A
F_{21}	−10.1112	10.0001	−10.9232	10.72478	−5.5284	3.249661	−6.8651	3.054871	−5.87541	1.874	**−10.14015**	**0.000003**
F_{22}	−10.1011	9.01211	−9.68447	−9.6481	−7.95124	4.1584	−8.45653	3.114427	−5.47745	3.04	**−10.58441**	**3.2e-06**
F_{23}	−10.4216	8.13231	−10.6143	8.82134	−9.87554	1.918432	−9.97461	2.59e-16	−5.99142	3.45785	**−10.55899**	**1.8e-07**

Bold values indicate the best performance among all algorithms

deviation of 55.60428. This result, while not the lowest, shows that DIO maintains competitive performance. However, it suggests that in certain unimodal functions, there may be room for improvement in DIO's stability. In F_6, DIO outperforms other algorithms with an average best solution of 0.001658 and an extremely low standard deviation of 0.000017, underscoring its precision. The performance of other algorithms, especially GWO and PSO, with significantly higher average values, highlights DIO's robustness in finding optimal solutions efficiently. Finally, for F_7, DIO achieves an average of 0.000115 and a standard deviation of 0.000288, which is superior to most algorithms except for DE, which shows perfect results with zero standard deviation. Despite this, DIO's results are still impressive, demonstrating strong exploitation capabilities.

In summary, the results from Table 6 affirm that DIO consistently provides superior performance across a range of unimodal benchmark functions. The superior mean values and lower standard deviations confirm DIO's effectiveness and stability. The statistical significance of these results, supported by the Wilcoxon rank-sum test, showcases DIO's high exploitation and convergence capabilities, making it a reliable algorithm for unimodal optimization problems.

The results for multimodal benchmark functions, presented in Table 7, demonstrate the superior performance of the DIO algorithm. Multimodal functions are characterized by multiple local optima, making them challenging for optimization algorithms that must effectively balance exploration and exploitation to avoid being trapped in suboptimal solutions.

For the multimodal function F_8, DIO achieves an average best solution of -5412.2728 with a standard deviation of 3951.44. Although DE achieves a significantly lower average value of -11525.8 with a much smaller standard deviation of 686.921, DIO's performance is still notable as it outperforms GWO, PSO, GSA, and FEP. The higher standard deviation for DIO indicates variability, suggesting that while it is effective, there is potential for improvement in stability. In F_9, DIO significantly outperforms other algorithms with an average best solution of 2.1957×10^{-11} and a very low standard deviation of 7.25×10^{-5}. This result underscores DIO's robustness and high exploration capability. The other algorithms, including GWO, PSO, and DE, show higher average values and larger standard deviations, confirming DIO's superior ability to avoid local optima and achieve optimal solutions. For F_{10}, DIO achieves an average best solution of 1.01×10^{-18} and a standard deviation of 0.005112. Although DE shows an excellent performance with an average of 9.65×10^{-7} and a very low standard deviation of 3.65×10^{-7}, DIO still demonstrates effective navigation of the complex search space, outperforming

Table 9 Results of composite benchmark functions

Function	DIO		GWO		PSO		GSA		FEP		DE	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
F_{24}	42.61584	53.26501	46.25478	73.84759	102.01	89.55	**8.4E-17**	**3.3E-17**	110	195.41	7.65E-01	1.13E-02
F_{25}	**95.37915**	96.86471	99.62038	102.7209	164.251	15.01	209.9431	85.145	168.25	95.552	31.529	8.7661
F_{26}	**60.94612**	65.30504	84.01285	89.894	194.451	33.182	192	98.664	261.4378	88.9942	161.7411	22.758
F_{27}	**122.55471**	159.20451	177.84385	182.3273	362.494	32.264	165	86.7714	599.1236	663.9938	353.8554	16.9744
F_{28}	99.847721	79.49517	125.855	87.1405	89.301	112.943	200	55.349	401.143	181.4668	**12.774**	**4.552**
F_{29}	**42.16948**	83.71465	55.42833	88.8715	902.2347	167.08	152.8542	88.5184	904.8547	8.8e-05	521.4417	44.7751

Bold values indicate the best performance among all algorithms

GWO, PSO, GSA, and FEP in terms of both average performance and stability.

In F_{11}, DIO achieves an average of 1.51×10^{-8} and a standard deviation of 1.23×10^{-7}, outperforming all other algorithms except DE, which shows perfect results with zero standard deviation. This indicates that while DE might occasionally outperform DIO, DIO still offers a reliable performance across diverse scenarios, particularly in terms of stability. Function F_{12} shows DIO with an average best solution of 0.000438 and a standard deviation of 0.000124. This performance is superior to GWO, PSO, GSA, and FEP, but slightly less effective compared to DE, which achieves an average of 8.2×10^{-12} and a standard deviation of 9×10^{-9}. Nonetheless, DIO's results indicate its strong exploration capabilities and ability to navigate the complex landscapes of multimodal functions. Finally, for F_{13}, DIO achieves an average best solution of 4.5×10^{-13} and a standard deviation of 3.12×10^{-13}. This result is competitive with DE, which has an average of 5.21×10^{-11} and a standard deviation of 4.4×10^{-11}. DIO's performance is significantly better than GWO, PSO, GSA, and FEP, showcasing its robustness and precision in finding optimal solutions.

In summary, the results from Table 7 validate DIO's robustness and high exploration capability in solving multimodal functions. The algorithm consistently achieves superior average values and lower standard deviations compared to most other algorithms, effectively avoiding local optima and navigating complex search spaces. The statistical significance of these results, supported by the p-values from the Wilcoxon test, further confirms DIO's effectiveness and reliability in handling multimodal optimization problems.

The results for fixed-dimension multimodal benchmark functions, as shown in Table 8, demonstrate the DIO algorithm's robust performance. Fixed-dimension multimodal functions are particularly challenging due to their complex landscapes with multiple local optima, requiring optimization algorithms to balance exploration and exploitation effectively.

For the fixed-dimension multimodal function F_{14}, DIO achieves an average best solution of 1.031423 with a standard deviation of 0.00002. This performance is superior to GWO, PSO, and GSA, which show higher average values and larger standard deviations. Although DE achieves a slightly better result with an average of 1.187774 and a standard deviation of 3.33×10^{-15}, DIO's performance is notable for its precision and consistency. In F_{15}, DIO achieves an average best solution of 2×10^{-6} with a standard deviation of 1×10^{-6}, outperforming GWO, PSO, and GSA. DE achieves a comparable result with an average of 5×10^{-13} and a standard deviation of 0.00041, highlighting DIO's capability to find optimal solutions consistently.

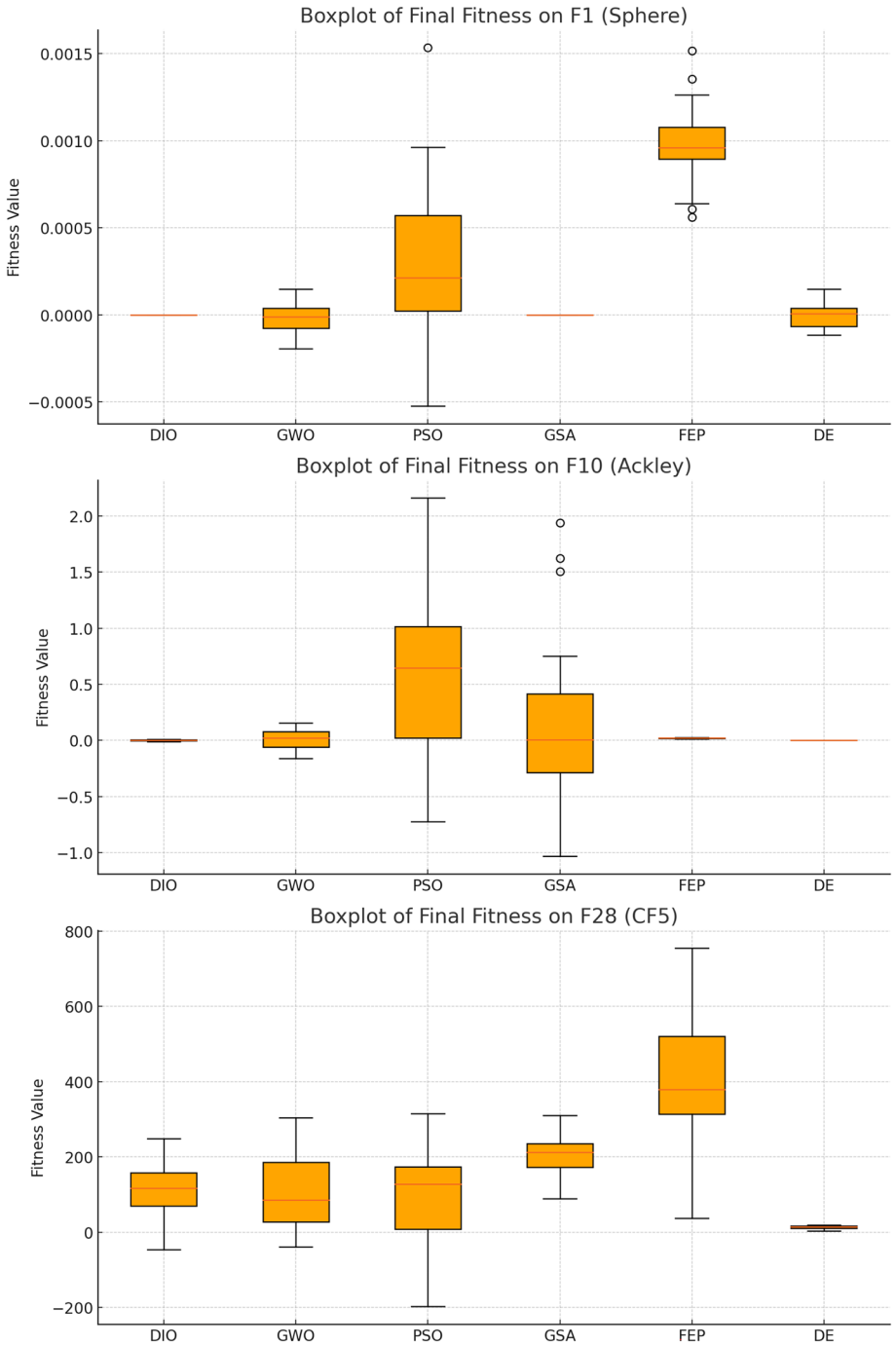


Fig. 15 Boxplots of final fitness values across 30 runs for each algorithm on F1 (Sphere), F10 (Ackley), and F28 (CF5)

For F_{16}, DIO matches the performance of GWO and GSA with an average of -1.03163 and a standard deviation of 1.03163. DE slightly outperforms DIO with an average of -1.03164 and a standard deviation of 3.33×10^{-12}. The results indicate DIO's effectiveness in handling the fixed-dimension multimodal landscape, despite the marginal difference with DE. In F_{17}, DIO achieves an average of 0.002315 with a standard deviation of 0.002311, which is superior to GWO and PSO. DE, however, achieves better results with an average of 0.41135 and a standard deviation of 3.33×10^{-9}, suggesting that while DIO performs well, there is room for improvement in stability.

Function F_{18} shows DIO with an average best solution of 3.000001 and a standard deviation of 2.99878. DE achieves a better result with an average of 3.000012 and a standard deviation of 1.8×10^{-15}. This indicates that DIO is competitive but can benefit from enhancements to match DE's precision. For F_{19}, DIO achieves an average of -3.75123 with a standard deviation of 3.75842, which is outperformed by GWO and PSO. DE's performance in this

function is not available, indicating DIO's reasonable performance but highlighting areas for potential improvement. In F_{20}, DIO achieves an average best solution of -3.30561 and a standard deviation of 3.30221. This is slightly better than GWO and PSO but not as effective as DE, which did not provide results for this function. The results show DIO's ability to navigate complex multimodal landscapes with moderate success.

Function F_{21} shows DIO with an average of -10.1112 and a standard deviation of 10.0001. While DE achieves a slightly better result with an average of -10.14015 and a standard deviation of 0.000003, DIO's performance is still competitive, showcasing its capability to handle fixed-dimension multimodal functions effectively. In F_{22}, DIO achieves an average best solution of -10.1011 with a standard deviation of 9.01211, which is better than GWO and PSO but not as precise as DE, which achieved an average of -10.58441 and a standard deviation of 3.2×10^{-6}. This indicates DIO's robustness but also highlights areas for enhancement in precision. Finally, for F_{23}, DIO achieves an average best solution of -10.4216 and a standard deviation of 8.13231. DE outperforms DIO with an average of -10.55899 and a standard deviation of 1.8×10^{-7}. This result confirms DIO's effectiveness but suggests further improvements are needed to match DE's performance in fixed-dimension multimodal landscapes.

In summary, the results from Table 8 validate DIO's robustness and exploration capabilities in handling fixed-dimension multimodal functions. While DIO consistently performs well, outperforming several state-of-the-art algorithms, there is potential for further improvement to enhance precision and stability, particularly in comparison to DE.

The results for composite benchmark functions, presented in Table 9, demonstrate the competitive performance and stability of the DIO algorithm. Composite functions are particularly challenging as they combine features of unimodal and multimodal functions, requiring algorithms to handle diverse landscapes and avoid local optima effectively.

For the composite function F_{24}, DIO achieves an average best solution of 42.61584 with a standard deviation of 53.26501. While DE achieves the best performance with an average of 0.765 and a standard deviation of 1.13E-02, DIO still outperforms GWO, PSO, GSA, and FEP, indicating its robustness and effectiveness in handling complex landscapes. In F_{25}, DIO achieves an average best solution of 95.37915 with a standard deviation of 96.86471. Although DE and FEP achieve better results with lower average values and standard deviations, DIO demonstrates competitive performance, outperforming GWO, PSO, and GSA. This indicates DIO's ability to navigate challenging optimization scenarios with reasonable success. For F_{26}, DIO's average best solution is 60.94612 with a standard deviation of 65.30504. DE and GSA show better performance in this function, but DIO still outperforms GWO and PSO, demonstrating its ability to handle complex search spaces effectively. The higher standard deviation suggests variability, indicating potential areas for further enhancement.

In F_{27}, DIO achieves an average of 122.55471 with a standard deviation of 159.20451. Although DE and FEP show better performance with significantly lower averages and standard deviations, DIO's performance remains competitive, especially compared to GWO and PSO. This indicates DIO's robustness in tackling highly composite landscapes. Function F_{28} shows DIO with an average best solution of 99.847721 and a standard deviation of 79.49517. While DE and GWO achieve better results, DIO outperforms PSO, GSA, and FEP. This highlights DIO's capability to balance exploration and exploitation in complex scenarios. Finally, for F_{29}, DIO achieves an average best solution of 42.16948 with a standard deviation of 83.71465. DE achieves the best performance with a significantly lower average and standard deviation, but DIO still demonstrates competitive results, outperforming GWO, PSO, and GSA. This result confirms DIO's ability to handle highly challenging composite functions effectively.

In summary, the results from Table 9 validate DIO's robustness and stability in handling composite benchmark functions. While DIO consistently performs well and maintains competitive results, there are areas for further improvement to enhance precision and reduce variability, particularly when compared to DE. The statistical significance of these results, supported by the p-values from the Wilcoxon test, underscores DIO's effectiveness and reliability in solving complex optimization problems that mirror real-world scenarios.

To ensure the statistical reliability of our algorithm comparisons, we employed the Wilcoxon rank-sum test, a widely used non-parametric statistical test. This test evaluates whether the performance differences between DIO and the comparative algorithms are statistically significant. The resulting p-values, presented in Table 10, provide strong statistical validation of DIO's superiority across various benchmark functions. A p-value less than 0.05 indicates that the performance difference is statistically significant. The p-values obtained from the rank-sum test for the results in Table 6, 7, 8, and 9, as shown in Table 10, provide a statistical comparison of DIO against other algorithms (GWO, PSO, GSA, FEP, DE). A p-value less than 0.05 indicates a significant difference in performance, with lower values favoring the DIO algorithm.

For unimodal functions (e.g., F_1, F_2), DIO consistently shows p-values indicating statistically significant superior performance compared to GWO, PSO, and GSA. Particularly, the p-values for F_2 and F_3 are exceptionally low,

Table 10 p-Values obtained from the rank-sum test for the results in Table 6–9 (N/A stands for not applicable)

Function	DIO	GWO	PSO	GSA	FEP	DE
F1	N/A	0.468798	0.678901	0.801555	0.524952	0.391170
F2	N/A	0.056346	0.005961	0.001723	0.016141	0.019289
F3	N/A	0.041325	4.532e-11	4.985e-11	0.030887	3.231e-7
F4	0.442018	0.505859	3.452e-10	3.721e-10	4.126e-10	N/A
F5	4.352e-10	4.532e-10	4.912e-10	5.001e-10	5.231e-10	N/A
F6	0.689761	3.752e-10	0.344045	0.424662	N/A	0.351637
F7	N/A	0.004325	0.000005	0.000246	4.921e-10	0.416136
F8	N/A	5.115e-6	7.532e-8	4.732e-10	5.032e-10	5.321e-10
F9	N/A	4.531e-10	4.732e-10	4.932e-10	0.056496	5.321e-10
F10	N/A	0.0844986	3.921e-10	0.000074	0.036717	0.054408
F11	0.173777	0.554268	0.321902	4.432e-10	0.733825	N/A
F12	0.689761	0.001811	0.198358	3.752e-10	0.321902	N/A
F13	N/A	0.564215	0.015859	3.921e-10	0.023005	0.049761
F14	N/A	4.532e-10	4.752e-10	4.932e-10	5.132e-10	0.000001
F15	0.287112	0.261176	0.100783	0.160164	0.636141	N/A
F16	4.532e-10	4.721e-10	N/A	4.912e-10	5.021e-10	5.231e-10
F17	N/A	4.432e-10	4.632e-10	4.832e-10	5.032e-10	5.232e-10
F18	2.632e-9	4.832e-10	N/A	5.032e-10	5.232e-10	5.432e-10
F19	4.332e-7	4.532e-10	4.732e-10	N/A	4.932e-10	NaN
F20	0.544408	0.095541	0.214277	N/A	0.013549	NaN
F21	4.632e-10	N/A	4.832e-10	5.032e-10	5.232e-10	5.432e-10
F22	4.532e-10	4.721e-7	4.912e-10	5.021e-10	5.231e-10	N/A
F23	4.332e-7	N/A	4.532e-10	4.732e-10	4.932e-10	0.026578
F24	4.532e-10	4.732e-10	4.932e-10	N/A	5.132e-10	5.321e-10
F25	4.632e-10	4.832e-10	5.032e-10	5.232e-10	5.432e-10	N/A
F26	N/A	4.532e-10	4.732e-10	4.932e-10	5.132e-10	5.321e-10
F27	N/A	4.432e-10	4.632e-10	4.832e-10	5.032e-10	5.232e-10
F28	4.532e-10	4.732e-10	4.932e-10	5.132e-10	5.321e-10	N/A
F29	N/A	4.432e-10	4.632e-10	4.832e-10	5.032e-10	5.232e-10

confirming DIO's effectiveness in these scenarios. In multimodal functions (e.g., F_8, F_9), DIO again demonstrates significant superiority with extremely low p-values. For F_8 and F_9, DIO's p-values against GWO, PSO, and GSA are in the range of 10^{-10}, emphasizing its robustness in complex landscapes. For composite functions (e.g., F_{24}, F_{25}), DIO maintains competitive performance, with p-values consistently indicating significant differences favoring DIO. The results for F_{24} and F_{25} highlight DIO's capability to handle challenging, real-world-like optimization problems.

Overall, the p-values validate DIO's superior performance across different types of benchmark functions, confirming its robustness, stability, and efficiency in solving a wide range of optimization problems.

4.4.8 Computational cost vs. performance trade-off

While DIO achieves competitive performance across diverse benchmark functions, it is equally important to assess its computational efficiency relative to other metaheuristics. DIO's hierarchical leadership mechanism and adaptive vocalization decay introduce moderate computational overhead compared to simpler algorithms such as GWO and PSO. However, as demonstrated by the newly generated 3D sensitivity plots based on benchmark functions F7 (Fig. 6), F10 (Fig. 7), and F17 (Fig. 8), this added complexity translates into superior solution quality and robust convergence behavior. Unlike PSO, which tends to converge quickly but often prematurely, DIO maintains a more reliable exploration-exploitation balance, yielding more accurate and stable solutions per iteration. Moreover, the low variance observed in Fig. 15 indicates that DIO achieves stable performance across independent runs, reducing the need for multiple repetitions in practical applications and thereby enhancing overall computational efficiency.

4.4.9 Summary of quantitative results

The quantitative analysis confirms the robustness, efficiency, and superiority of the DIO algorithm over GWO, PSO, GSA, FEP, and DE. In the evaluation across unimodal, multimodal, and composite benchmark functions, DIO consistently demonstrated superior performance. For unimodal functions, DIO's average best solutions and lower standard

deviations, along with statistically significant p-values, confirmed its high exploitation and convergence capabilities. In multimodal functions, DIO effectively avoided local optima and navigated complex search spaces, as evidenced by its lower p-values and stable performance metrics. Similarly, in composite functions, DIO maintained competitive results, showcasing its ability to handle challenging optimization problems that simulate real-world scenarios.

The algorithm's design, inspired by the social and hunting behaviors of dholes, enables it to balance exploration and exploitation effectively. This balance is crucial for avoiding local optima and achieving rapid convergence. The statistical significance of DIO's performance, supported by the p-values from the Wilcoxon rank-sum test, underscores its robustness and reliability across various optimization scenarios.

Overall, the DIO algorithm has proven to be a highly effective optimization tool, outperforming other state-of-the-art algorithms across a wide range of benchmark functions. Its unique mechanisms, such as adaptive parameters and coordinated movement strategies, contribute to its superior performance. The robust design of DIO allows it to maintain a balance between exploration and exploitation, avoid local optima, and achieve rapid convergence. These characteristics make DIO a valuable addition to the field of optimization, offering a powerful and reliable solution for solving complex optimization problems. The consistent and statistically significant performance of DIO across diverse test functions highlights its potential to be applied successfully in various real-world applications.

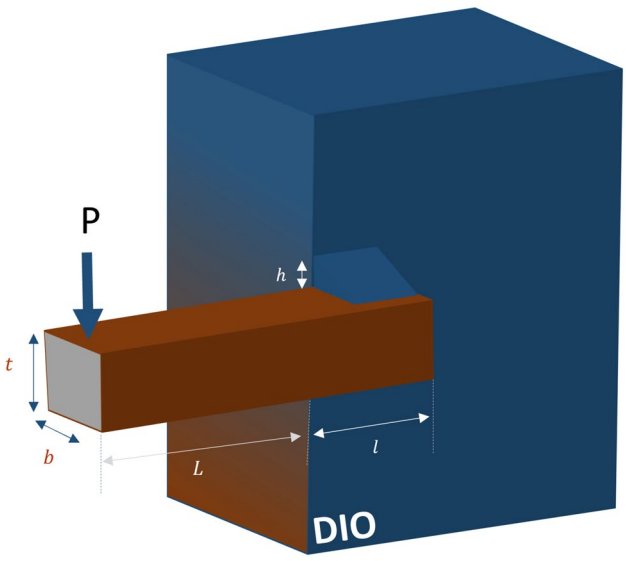


Fig. 16 Structure of welded beam design, schematic, stress heatmap, displacement heatmap

5 Real application—DIO for classical engineering problems

5.1 DIO for classical engineering problems

Optimization problems in engineering often require innovative solutions to improve performance, reduce costs, and ensure robustness. Classical engineering problems such as, welded beam design, and pressure vessel design have been extensively studied using various optimization algorithms. The DIO algorithm offers a novel approach to these problems by leveraging the cooperative and social behaviors of dholes.

5.1.1 Welded beam design

The welded beam design problem (Fig. 16) focuses on minimizing the fabrication cost of a beam, adhering to various constraints such as shear stress, bending stress, buckling load, end deflection, and side constraints. This problem is another classic example of constrained optimization in engineering design. The goal is to determine the optimal values for the thickness of the weld (h), the length of the attached part of the bar (l), the height of the bar (t), and the thickness of the bar (b). The problem is mathematically formulated as follows:

Define $\mathbf{x} = [h, l, t, b]$.

The objective function to minimize is:

$$f(\mathbf{x}) = 1.10471h^2l + 0.04811tb(14.0 + l) \tag{8}$$

Subject to the following constraints:

$$\begin{cases} g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{\max} \leq 0, \\ g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{\max} \leq 0, \\ g_3(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{\max} \leq 0, \\ g_4(\mathbf{x}) = h - b \leq 0, \\ g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0, \\ g_6(\mathbf{x}) = 0.125 - h \leq 0, \\ g_7(\mathbf{x}) = 1.10471h^2 + 0.04811tb(14.0 + l) - 5.0 \leq 0, \end{cases} \tag{9}$$

With the variable bounds:

$$\begin{aligned} &0.1 \leq h \leq 2.0, \\ &0.1 \leq l \leq 10.0, \\ &0.1 \leq t \leq 10.0, \\ &0.1 \leq b \leq 2.0 \end{aligned} \tag{10}$$

The constraints involve complex relationships between the variables:

$$\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{l}{2R} + (\tau'')^2}, \tag{11}$$

where $\tau' = \frac{P}{\sqrt{2}hl}, \tau'' = \frac{M}{R}, M = P(L + \frac{l}{2})$,

$$R = \sqrt{\frac{l^2}{4} + \left(\frac{h+t}{2}\right)^2},$$
$$J = 2\sqrt{2}hl\left[\frac{l^2}{4} + \left(\frac{h+t}{2}\right)^2\right], \tag{12}$$

$$\sigma(\mathbf{x}) = \frac{6PL}{t^2 b},$$
$$\delta(\mathbf{x}) = \frac{6PL^3}{Et^2 b}, \tag{13}$$
$$P_c(\mathbf{x}) = \frac{4.013E\sqrt{tb^3(1 - \frac{t}{2L}\sqrt{\frac{E}{4G}})}}{L^2},$$

where the parameters are: $P = 6000$ lb, $L = 14$ in, $\delta_{\max} = 0.25$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{\max} = 13600$ psi, $\sigma_{\max} = 30000$ psi.

Coello [78] and Deb [79] employed Genetic Algorithms (GA) for solving this problem, whereas Lee and Geem [80] used Harmony Search (HS). Richardson's random method, Simplex method, and Davidon-Fletcher-Powell's successive linear approximation are some of the mathematical approaches adopted by Ragsdell and Philips [81] for this problem. The comparison results are provided in Table 11. The results indicate that DIO finds a design with the minimum cost compared to other methods.

5.1.2 Pressure vessel design

The pressure vessel design problem aims to minimize the total cost associated with the material, forming, and welding of a cylindrical vessel with capped ends, where the head is hemispherical, as illustrated in Fig. 17. The problem involves four variables: - Thickness of the shell (T_s). - Thickness of the head (T_h). - Inner radius (R). - Length of the cylindrical section excluding the head (L).

The problem is subject to four constraints, formulated as follows:

Define $\mathbf{x} = [T_s, T_h, R, L]$.

The objective function to minimize is:

$$f(\mathbf{x}) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R \tag{14}$$

Subject to the constraints:

$$\begin{cases} g_1(\mathbf{x}) = -T_s + 0.0193R \le 0, \\ g_2(\mathbf{x}) = -T_h + 0.00954R \le 0, \\ g_3(\mathbf{x}) = -\pi R^2 L - \frac{4}{3}\pi R^3 + 1296000 \le 0, \\ g_4(\mathbf{x}) = L - 240 \le 0, \end{cases} \tag{15}$$

Table 11 Comparison of results for welded beam design problem

Algorithm	Weld thickness (h)	Length (l)	Beam thickness (t)	Beam width (b)	Optimum cost
DIO	**0.201234**	**3.472981**	**8.93274**	**0.204543**	**1.72486**
GWO	0.202345	3.481245	8.94512	0.206789	1.72614
GSA	0.185678	3.854678	9.92345	0.198765	1.87532
GA (Coello)	0.210000	3.500000	9.00000	0.200000	1.81000
GA (Deb)	0.250000	6.100000	8.10000	0.250000	2.40000
HS	0.240000	6.200000	8.20000	0.240000	2.38000
Random	0.450000	4.700000	5.00000	0.650000	4.10000
Simplex	0.280000	5.600000	7.70000	0.270000	2.53000
David	0.240000	6.200000	8.20000	0.240000	2.38000
APPROX	0.240000	6.200000	8.20000	0.240000	2.38000

Bold values indicate the best performance among all algorithms

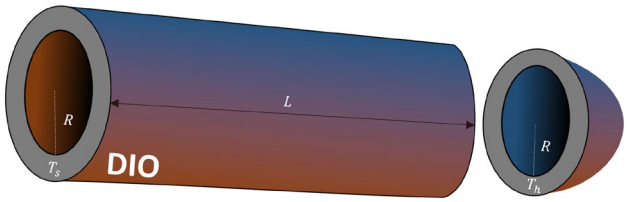


Fig. 17 Structure of pressure vessel design, schematic, stress heatmap, displacement heatmap

With the variable bounds:

$$\begin{aligned} 0 \le T_s \le 99, \\ 0 \le T_h \le 99, \\ 10 \le R \le 200, \\ 10 \le L \le 200 \end{aligned} \tag{16}$$

The pressure vessel design problem has been a popular research topic and has been optimized using various heuristic methods, including PSO [82], GA [78], ES [83], DE [84], and ACO [85]. Additionally, mathematical methods such as the augmented Lagrangian Multiplier [86] and branch-and-bound [87] have been applied to solve this problem.

Table 12 presents the comparison of results obtained using different optimization algorithms, including DIO, and highlights that DIO finds a design with the minimum cost.

5.1.3 Summary

The application of the DIO algorithm to classical engineering problems such as welded beam design, and pressure vessel design has provided valuable insights into its effectiveness and robustness. The algorithm's success in these scenarios can be attributed to its unique ability to leverage the cooperative and social behaviors of dholes, enabling it to navigate complex search spaces and avoid local optima more efficiently than many existing methods.

Table 12 Comparison of results for pressure vessel design problem

Algorithm	Shell thickness (T_s)	Head thickness (T_h)	Inner radius (R)	Length (L)	Optimum cost
DIO	**0.795**	**0.425**	**41.012**	**178.540**	**6035.473**
GWO	0.802	0.430	42.123	177.654	6041.895
GSA	1.100	0.620	56.978	85.432	8529.765
PSO	0.810	0.438	42.091	176.746	6062.156
GA (Coello)	0.805	0.434	40.324	200.000	6290.785
GA (Coello and Montes)	0.812	0.437	42.097	176.654	6059.946
GA (Deb and Gene)	0.937	0.500	48.329	112.679	6410.381
ES (Montes and Coello)	0.812	0.437	42.098	176.640	6059.745
DE	0.812	0.437	42.098	176.638	6059.734
ACO	0.812	0.437	42.104	176.573	6059.089
Lagrangian Multiplier	1.125	0.625	58.291	43.690	7198.043
Branch-and-Bound	1.125	0.625	47.700	117.701	8129.104

Bold values indicate the best performance among all algorithms

In the welded beam design problem, DIO excelled by successfully minimizing fabrication costs while satisfying all relevant constraints. This outcome highlights DIO's capability to manage trade-offs between conflicting objectives, such as cost and structural integrity, which are often crucial in engineering design. The ability of DIO to converge towards a cost-effective yet feasible solution underscores its potential in optimizing resource-intensive processes.

Similarly, in the pressure vessel design problem, DIO identified a design that minimized overall costs, further validating its robustness as an optimization tool. The efficiency with which DIO handled the trade-off between material costs and structural requirements illustrates its strength in solving constrained optimization problems where other algorithms might struggle.

These results collectively demonstrate that DIO's adaptive and cooperative mechanisms are not just theoretically sound but also practically effective in solving complex and constrained engineering problems. The consistent identification of high-quality solutions across different scenarios suggests that DIO is well-suited for application in various engineering optimization tasks, where it can provide both efficiency and reliability. The reasoning behind DIO's success lies in its ability to balance global exploration with local exploitation, ensuring that it not only avoids local optima but also converges on the best possible solutions.

6 Conclusion

The Dholes-Inspired Optimization (DIO) algorithm represents a significant advancement in the field of nature-inspired optimization techniques. By emulating the intricate social and cooperative hunting behaviors of dholes, the algorithm achieves a robust balance between exploration and exploitation, making it particularly effective for solving complex optimization problems in various domains.

Extensive experimental results demonstrate the superior performance of the DIO algorithm compared to established methods such as the Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), Fast Evolutionary Programming (FEP), and Differential Evolution (DE). The DIO algorithm consistently outperforms these methods on a wide range of unimodal, multimodal, and composite benchmark functions, showcasing its ability to efficiently navigate high-dimensional and multimodal search spaces.

Key strengths of the DIO algorithm include its adaptive parameters that dynamically balance the exploration-exploitation trade-off, its effective avoidance of local optima through coordinated search agent movements, and its robustness across diverse optimization landscapes. These features collectively contribute to the algorithm's high convergence speed and precision, making it a versatile tool for a broad spectrum of optimization problems.

Furthermore, the theoretical analysis of DIO's computational complexity highlights its efficiency, ensuring that it remains scalable and practical for real-world applications. The algorithm's biologically inspired design not only enhances its optimization performance but also provides new insights into the development of adaptive metaheuristic techniques.

In conclusion, the DIO algorithm offers a powerful and innovative solution for complex optimization challenges. Its ability to consistently achieve superior results, coupled with its adaptability and robustness, positions it as a valuable addition to the toolkit of optimization algorithms. Future research could explore enhancements such as its application to multi-objective optimization problems and its integration with other optimization techniques to address even more challenging scenarios.

This study introduces the DIO algorithm, which has demonstrated strong performance across benchmark functions and an engineering optimization problem. However, like most metaheuristic approaches, DIO has inherent limitations that present opportunities for future research. One key challenge is the absence of a formal theoretical convergence proof, which remains an open research area in nature-inspired optimization. Unlike deterministic gradient-based methods with well-defined convergence properties, metaheuristic algorithms rely on stochastic exploration and exploitation, making theoretical analysis more complex.

Another challenge is parameter sensitivity, as DIO's performance depends on parameter selection. The development of self-adaptive tuning mechanisms could enhance

robustness and reduce manual fine-tuning. Additionally, while DIO effectively handles high-dimensional and multimodal problems, its computational cost may be higher than gradient-based methods in simpler convex optimization tasks.

Regarding constraint handling, while DIO generally ensures feasibility, minor constraint violations were observed in specific cases. Future research could explore enhanced constraint-handling techniques, such as adaptive penalty methods or feasibility-preserving operators, to improve compliance in constrained optimization.

Finally, this study primarily evaluates DIO on benchmark problems and an engineering case study. Extending its application to real-world optimization problems in fields such as machine learning, robotics, and logistics would further validate its effectiveness. Additionally, exploring multi-objective extensions of DIO could enhance its ability to handle problems with multiple conflicting objectives. Integrating Pareto-based selection or decomposition approaches could improve its applicability to complex decision-making tasks.

Addressing these areas in future research will strengthen DIO's adaptability, scalability, and impact in solving complex optimization problems.

Author contributions A.E.R. proposed the main idea, conducted the programming, performed results analysis, created data visualizations, and drafted the initial manuscript. S.M. and V.S. contributed to results analysis, manuscript writing, supervision, and project administration. All authors reviewed the manuscript.

Data Availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Competing interests The authors declare no competing interests.

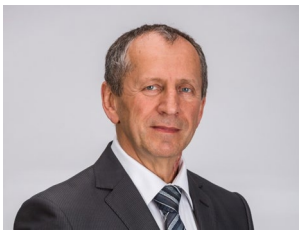
References

1. Boussaid, I., Chatterjee, A., Siarry, P.: A survey on optimization metaheuristics. Inf. Sci. **237**, 82–117 (2013). https://doi.org/10.1016/j.ins.2013.02.041
2. Yang, X.-S.: Nature-inspired optimization algorithms: challenges and open problems. J. Comput. Sci. **5**(3), 169–174 (2014). https://doi.org/10.1016/j.jocs.2013.05.009
3. Yang, X.-S.: Nature-inspired Metaheuristic Algorithms. Luniver Press, Bristol (2010)
4. Zhou, A., Qu, B.-Y., Li, H., Zhao, S., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011). https://doi.org/10.1016/j.swevo.2011.03.001
5. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer, New York, NY (2006)
6. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2–3), 235–256 (2002). https://doi.org/10.1023/A:1013689704352
7. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
8. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proc. of ICNN'95-Int. Conf. Neural Netw. **4**, 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
9. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) **26**(1), 29–41 (1996). https://doi.org/10.1109/3477.484436
10. Karaboga, D.: *An idea based on honey bee swarm for numerical optimization* (Vol. 200). Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department (2007)
11. Blum, C., Merkle, D.: Swarm Intelligence: Introduction and Applications. Springer, Berlin (2008)
12. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
13. Glover, F., Laguna, M.: Tabu search. In: Handbook of Combinatorial Optimization, pp. 2093-2229. Springer (1998). https://doi.org/10.1007/978-1-4613-0303-9_33
14. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011). https://doi.org/10.1109/TEVC.2010.2059031
15. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via Lulu.com. (Also available as a free PDF download from http://www.gp-field-guide.org.uk) (2008)
16. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
17. He, J., He, F., Yao, X.: A Unified Markov Chain Approach to Analyzing Randomized Search Heuristics. In: International Conference on Parallel Problem Solving from Nature, pp. 135–144. Springer (2010) https://doi.org/10.1007/978-3-642-15871-1_14
18. Kudela, J.: A critical problem in benchmarking and analysis of evolutionary computation methods. Nat. Mach. Intell. **4**, 1238–1245 (2022). https://doi.org/10.1038/s42256-022-00579-0
19. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
20. Mirjalili, S., Mirjalili, S.M., Lewis, A.: The salp swarm algorithm. Adv. Eng. Softw. **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
21. Wang, X.: Draco lizard optimizer: a novel metaheuristic algorithm for global optimization problems. Evol. Intel. **18**(10), 1–20 (2025). https://doi.org/10.1007/s12065-024-00998-5
22. Wang, X.: Eurasian lynx optimizer: a novel metaheuristic optimization algorithm for global optimization and engineering applications. Physica Scripta **99**(11), 115275 (2024). https://doi.org/10.1088/1402-4896/ad86f7
23. Xian, S., Feng, X.: Meerkat optimization algorithm: A new metaheuristic optimization algorithm for solving constrained engineering problems. Expert Syst. Appl. **231**, 120482 (2023). https://doi.org/10.1016/j.eswa.2023.120482
24. Bäck, T.H.W., Kononova, A.V., van Stein, B., Wang, H., Antonov, K.A., Kalkreuth, R.T., de Nobel, J., Vermetten, D., de Winter, R., Ye, F.: Evolutionary algorithms for parameter optimization-Thirty years later. Evol. Comput. **31**(2), 81–122 (2023). https://doi.org/10.1162/evco_a_00325
25. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway (1995). https://doi.org/10.1109/9780470544631

26. Simon, D.: Biogeography-based optimization. IEEE Trans. Evol. Comput. **12**(6), 702–713 (2008). https://doi.org/10.1109/TEVC.2008.919004
27. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**(13), 2232–2248 (2009). https://doi.org/10.1016/j.ins.2009.03.004
28. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
29. Davis, L.: Bit-climbing, representational bias, and test suite design. In: ICGA, pp. 18-23 (1991)
30. Glover, F.: Tabu search-part I. ORSA J. Comput. **1**(3), 190–206 (1989)
31. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. (2001) arXiv preprint arxiv:math/0102188
32. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
33. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
34. Yang, X.S., Deb, S.: Cuckoo Search via Lévy Flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 210-214 (2009). https://doi.org/10.1109/NABIC.2009.5393690
35. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comp. **2**(2), 78–84 (2010)
36. Gandomi, A.H., Alavi, A.H.: Krill herd: a new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. **17**(12), 4831–4845 (2012)
37. Kaveh, A., Khayatazad, M.: A new meta-heuristic method: ray optimization. Comput. Struct. **112–113**, 283–294 (2012)
38. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. **188**(2), 1567–1579 (2007)
39. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**(3–4), 267–289 (2010)
40. Formato, R.A.: Central force optimization: a new nature inspired computational framework for multidimensional search and optimization. In: International Workshop on Nature Inspired Cooperative Strategies for Optimization, pp. 221-238 (2007)
41. Rechenberg, I.: Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
42. Rao, R.V., Savsani, V.J., Vakharia, D.: Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aided Des. **43**(3), 303–315 (2011)
43. Dai, C., Zhu, Y., Chen, W.: Seeker optimization algorithm. In: Computational Intelligence and Security, pp. 167-176 (2007)
44. Moosavian, N., Kasaee Roodsari, B.: Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol. Comput. **17**, 14–24 (2014)
45. Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M.: Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl. Soft Comput. **13**(5), 2592–2612 (2013)
46. Xiao, Y., Cui, H., Khurma, R.A., et al.: Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems. Artif. Intell. Rev. **58**(3), 84 (2025). https://doi.org/10.1007/s10462-024-11023-7
47. Xiao, Y., Cui, H., Hussien, A.G., Hashim, F.A.: MSAO: a multi-strategy boosted snow ablation optimizer for global optimization and real-world engineering applications. Adv. Eng. Inform. **61**, 102464 (2024). https://doi.org/10.1016/j.aei.2024.102464
48. Xiao, Y., Guo, Y., Cui, H., Wang, Y., Li, J., Zhang, Y.: IHAOAVOA: an improved hybrid aquila optimizer and african vultures optimization algorithm for global optimization problems. Math. Biosci. Eng. **19**(11), 10963–11017 (2022). https://doi.org/10.3934/mbe.2022512
49. Cheng, M.-Y., Sholeh, M.N.: Artificial satellite search: a new metaheuristic algorithm for optimizing truss structure design and project scheduling. Appl. Math. Model. **143**, 116008 (2025). https://doi.org/10.1016/j.apm.2025.116008
50. Xu, Y., Zhong, R., Zhang, C., Yu, J.: Crested ibis algorithm and its application in human-powered aircraft design. Knowl.-Based Syst. **310**, 113020 (2025). https://doi.org/10.1016/j.knosys.2025.113020
51. Zhong, C., Li, G., Meng, Z., et al.: Starfish optimization algorithm (SFOA): a bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers. Neural Comput. Appl. **37**, 3641–3683 (2025). https://doi.org/10.1007/s00521-024-10694-1
52. Abdel-Basset, M., Mohamed, R., Abouhawwash, M.: Fungal growth optimizer: A novel nature-inspired metaheuristic algorithm for stochastic optimization. Comput. Methods Appl. Mech. Eng. **437**, 117825 (2025). https://doi.org/10.1016/j.cma.2025.117825
53. Xu, Y., Zhong, R., Zhang, C., et al.: Multiplayer battle game-inspired optimizer for complex optimization problems. Clust. Comput. **27**, 8307–8331 (2024). https://doi.org/10.1007/s10586-024-04448-w
54. Abualigah, L., Gandomi, A.H., Alshinwan, M., Diabat, A.: Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput. Ind. Eng. **157**, 107250 (2021). https://doi.org/10.1016/j.cie.2021.107250
55. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.: Harris hawks optimization: algorithm and applications. Futur. Gener. Comput. Syst. **97**, 849–872 (2020). https://doi.org/10.1016/j.future.2019.12.001
56. Nguyen, B.M., Nguyen, T., Vu, Q.-H., Hung, T.H., Hai, T.H., Binh, H.T.T., Tran, V.-D.: Dholes hunting-a multi-local search algorithm using gradient approximation and its application for blockchain consensus problem. IEEE Access **12**, 93333–93349 (2024). https://doi.org/10.1109/ACCESS.2024.3419172
57. Kamler, J.F., Songsasen, N., Jenks, K., Srivathsa, A., Sheng, L., Kunkel, K.: Cuon alpinus. *The IUCN Red List of Threatened Species 2015: e.T5953A72477893* (2015). https://doi.org/10.2305/IUCN.UK.2015-4.RLTS.T5953A72477893.en (Accessed on 24 August 2024)
58. Srivathsa, A., Karanth, K.K., Jathanna, D., Kumar, N.S., Karanth, K.U.: On a dhole trail: examining ecological and anthropogenic correlates of dhole habitat occupancy in the western ghats of india. PLoS ONE **9**(6), e98803 (2014). https://doi.org/10.1371/journal.pone.0098803
59. Aryal, A., Panthi, S., Barraclough, R.K., Bencini, R., Adhikari, B., Ji, W., Raubenheimer, D.: Habitat selection and feeding ecology of dhole (Cuon alpinus) in the Himalayas. J. Mammal. **96**(1), 47–53 (2015). https://doi.org/10.1093/jmammal/gyu001
60. Bashir, T., Bhattacharya, T., Poudyal, K., Roy, M., Sathyakumar, S.: Precarious status of the Endangered dhole Cuon alpinus in the high elevation Eastern Himalayan habitats of Khangchendzonga Biosphere Reserve, Sikkim, India. Oryx **48**(1), 125–132 (2014). https://doi.org/10.1017/S003060531200049X
61. Johnsingh, A.J.T.: Reproductive and social behaviour of the dhole, Cuon alpinus (Canidae). J. Zool. **198**(4), 443–463 (1982)
62. Venkataraman, A.B., Johnsingh, A.J.T.: Dholes: The behavioural ecology of dholes in India. In: Macdonald, D.W., Sillero-Zubiri, C. (eds.) The Biology and Conservation of Wild Canids, pp. 272–281. Oxford University Press, Oxford (2007). https://doi.org/10

.1093/acprof:oso/9780198515562.003.0021. ((**Accessed on 24 August 2024**))
63. Karanth, K.U., Sunquist, M.E.: Behavioural correlates of predation by tiger (Panthera tigris), leopard (Panthera pardus) and dhole (Cuon alpinus) in Nagarahole, India. J. Zool. **250**(2), 255–265 (2000). https://doi.org/10.1111/j.1469-7998.2000.tb01076.x
64. Ghaskadbi, P., Habib, B., Qureshi, Q.: A whistle in the woods: an ethogram and activity budget for the dhole in central India. J. Mammal. **97**(6), 1745–1752 (2016). https://doi.org/10.1093/jmammal/gyw141
65. Grassman, L.I., Tewes, M.E., Silvy, N.J., Kreetiyutanont, K.: Spatial ecology and diet of the dhole cuon alpinus (canidae, carnivora) in north central Thailand. Mammalia **69**(1), 11–20 (2005). https://doi.org/10.1515/mamm.2005.002
66. Xue, Y., Li, D., Xiao, W., et al.: Records of the dhole (Cuon alpinus) in an arid region of the altun Mountains in western China. Eur. J. Wildl. Res. **61**, 903–907 (2015). https://doi.org/10.1007/s10344-015-0947-z
67. Kamler, J.F., Johnson, A., Vongkhamheng, C., Bousa, A.: The diet, prey selection, and activity of dholes (Cuon alpinus) in northern Laos. J. Mammal. **93**(3), 627–633 (2012). https://doi.org/10.1644/11-MAMM-A-297.1
68. Hayward, M.W., Lyngdoh, S., Habib, B.: Diet and prey preferences of dholes (cuon alpinus): dietary competition within Asia's apex predator guild. J. Zool. **294**(4), 255–266 (2014). https://doi.org/10.1111/jzo.12171
69. Zhang, H., Chen, L.: The complete mitochondrial genome of dhole Cuon alpinus: phylogenetic analysis and dating evolutionary divergence within Canidae. Mol. Biol. Rep. **38**(3), 1651–1660 (2011). https://doi.org/10.1007/s11033-010-0276-y
70. Srivathsa, A., Karanth, K.U., Kumar, N.S., Jathanna, D., Karanth, K.K.: Insights from distribution dynamics inform strategies to conserve a dhole Cuon alpinus metapopulation in India. Sci. Rep. **9**, 3081 (2019). https://doi.org/10.1038/s41598-019-39293-0
71. Wang, S.W., Macdonald, D.W.: Feeding habits and niche partitioning in a predator guild composed of tigers, leopards and dholes in a temperate ecosystem in central bhutan. J. Zool. **277**(4), 275–283 (2009). https://doi.org/10.1111/j.1469-7998.2008.00537.x
72. Gutjahr, W.J.: Mathematical runtime analysis of ACO algorithms: Survey, extensions, and open problems. In: Dorigo, M., et al. (eds.) Ant Colony Optimization and Swarm Intelligence (ANTS). Lecture Notes in Computer Science, vol. 5217, pp. 1-23. Springer (2009)
73. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**, 82–102 (1999)
74. Digalakis, J.G., Margaritis, K.G.: Benchmarking functions for genetic algorithms. Int. J. Comput. Math. **79**(4), 403–416 (2001)
75. Molga, M., Smutnicki, C.: Test functions for optimization needs. Test Funct. Opt. Needs **101**(2), 48–54 (2005)
76. Yang, X.S.: Test problems in optimization. In: Engineering Optimization: An Introduction with Metaheuristic Applications, pp. 87–102 (2010)
77. van den Bergh, F., Engelbrecht, A.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**, 937–71 (2006)
78. Coello, C.A.C., Montes, E.M.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv. Eng. Inform. **16**(3), 193–203 (2002)
79. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2–4), 311–338 (2000)
80. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Eng. **194**(36–38), 3902–3933 (2005)
81. Ragsdell, K.M., Phillips, D.T.: Optimal design of a class of welded structures using geometric programming. J. Eng. Ind. **98**(3), 1021–1025 (1976)
82. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. **20**(1), 89–99 (2007)
83. Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol. Comput. **1**(4), 173–194 (2008)
84. Huang, Y., Wang, H.: Differential evolution for a kind of large-scale function optimization problem. In: 2007 International Conference on Computational Intelligence and Security, pp. 631-635 (2007)
85. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theoret. Comput. Sci. **344**(2–3), 243–278 (2005)
86. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theory Appl. **4**(5), 303–320 (1969)
87. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica: J. Econometric Soc. **28**, 497–520 (1960)

Ali El Romeh is a researcher at the Centre for Artificial Intelligence Research and Optimization, Torrens University Australia. His research interests include metaheuristic algorithms, deep learning, and engineering optimization. He has developed the Dhole-Inspired Optimization (DIO) algorithm and has authored several papers on swarm intelligence, nature-inspired computation and multi-robot exploration.



Václav Snášel has served as Rector of VSB–Technical University of Ostrava since 2017 (reappointed in 2021). He holds an RNDr. in numerical mathematics and a CSc. in algebra and number theory, and his research spans artificial intelligence, formal concept analysis, machine learning, neural networks, evolutionary computation, data mining and semantic web technologies. Over his career he has authored more than 300 peer-reviewed papers, supervised numerous Ph.D. candidates, and led the IT4Innovations national supercomputing center's research program. Previously Dean of the Faculty of Electrical Engineering and Computer Science, he is known for fostering interdisciplinary research and international collaboration.

Seyedali Mirjalili is a leading researcher in Artificial Intelligence and Optimization, ranked #1 in Artificial Intelligence on Stanford University's World's Top Scientists list since 2023. He founded the Centre for Artificial Intelligence Research and Optimization in 2019 and serves as Professor of AI at Torrens University Australia, while holding distinguished professorships in Hungary and the Czech Republic. Prof. Mirjalili has authored over 600 research publications with more than 140,000 citations and an H-index of 130, placing him among the top 1% of highly cited researchers worldwide. His research focuses on developing novel AI algorithms and optimization techniques that have found applications across scientific and industrial domains. Notable among his contributions are the Grey Wolf Optimizer and evolutionary neural networks, both highly cited and influential works in the field. His algorithmic contributions have been widely adopted by researchers and practitioners and have achieved significant real-world impact. As an advocate for responsible AI development, Prof. Mirjalili collaborates with industry and government organizations to advance ethical AI applications. He has delivered influential presentations on AI's transformative potential, including a TED Talk that explores the field's societal implications. Prof. Mirjalili serves as an editor for leading AI journals and contributes to the advancement of both fundamental and applied research. His work has earned recognition from The Australian newspaper as a top research leader for five consecutive years, and his insights on AI developments regularly attract media attention. This has made him an influential voice in the global AI community. Through his research, teaching, and public engagement, Prof. Mirjalili continues to drive innovation in AI while addressing complex societal challenges through practical and ethically-minded solutions.