

Nubian Ibex Optimizer: A Nature-inspired Metaheuristic Algorithm

Mohammed Gamal Ragab

`mohammed.ragab@utp.edu.my`

Universiti Teknologi Petronas <https://orcid.org/0000-0001-5226-4962>

Said Jadid Abdulkadir

Universiti Teknologi Petronas <https://orcid.org/0000-0003-0038-3702>

Hitham Alhussian

Universiti Teknologi Petronas <https://orcid.org/0000-0003-3947-269X>

Alawi Alqushaibi

University of Nottingham Malaysia Campus <https://orcid.org/0000-0002-3001-1224>

Ebrahim Hamid Sumiea

Universiti Teknologi Petronas <https://orcid.org/0000-0002-2829-0842>

Safwan Mahmood Al-Selwi

SDAIA-KFUPM Joint Research Center for Artificial Intelligence, King Fahd University of Petroleum & Minerals <https://orcid.org/0009-0000-2742-213X>

Amgad Muneer

`muneeramgad@gmail.com`

Universiti Teknologi Petronas <https://orcid.org/0000-0002-7157-3020>

Ahmed Omer Ahmed Ismail

King Khalid University

Jia Wu

The University of Texas MD Anderson Cancer Center <https://orcid.org/0000-0001-8392-8338>

Research Article

Keywords: benchmark functions, engineering design optimization, meta-heuristic, nature-inspired algorithm, nubian ibex optimizer, optimization

Posted Date: October 13th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11294841/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Nubian Ibex Optimizer: A Nature-inspired Metaheuristic Algorithm

Mohammed Gamal Ragab ^{1,2 *}, Said Jadid Abdulkadir ^{1,2}, Hitham Alhussian^{1,2},
Alawi Alqushaibi ³, Ebrahim Hamid Sumiea ^{1,2}, Safwan Mahmood Al-Selwi ⁴,
Amgad Muneer ⁵, Ahmed Omer Ahmed Ismail⁶, and Jia Wu ⁵

¹Department of Computing, Universiti Teknologi PETRONAS, Seri Iskandar,
32610, Perak, Malaysia.

²Center for Research in Data Science (CeRDaS), Universiti Teknologi PETRONAS,
Seri Iskandar, 32610, Perak, Malaysia.

³School of Computer and Mathematical Sciences, University of Nottingham, Jalan
Broga, Semenyih, 43500, Selangor, Malaysia.

⁴SDAIA-KFUPM Joint Research Center for Artificial Intelligence, King Fahd
University of Petroleum & Minerals, Dhahran, 31261, Saudi Arabia

⁵Department of Imaging Physics, The University of Texas MD Anderson Cancer
Center, Houston, 78712, TX, USA.

⁶Unit of Specializations of Technology and Engineering, Applied College, King
Khalid University, Muhayil, 61913, Saudi Arabia.

*Address correspondence to: mohammed.ragab@utp.edu.my

Abstract

This paper introduces the Nubian Ibex Optimizer (NIO), a novel meta-heuristic inspired by the Nubian ibex's adept exploration and exploitation of challenging terrains. NIO effectively navigates complex search spaces, employing dynamic parameter adjustments and diversification strategies to avoid local optima and adapt to diverse optimization landscapes. The algorithm's performance is assessed on 23 benchmark functions, showing superior exploitation on unimodal functions and robust exploration on multimodal functions. Its efficacy is further evaluated through 8 engineering design problems (tension/compression spring, pressure vessel, speed reducer, three bar truss, welded beam, cantilever beam, multiple-disc clutch brake, piston lever), highlighting its potential in real-world challenges. Extensive experiments comparing NIO to other state-of-the-art algorithms demonstrated improved convergence rates and robustness across diverse optimization challenges. The Python NIO codes are available on GitHub, <https://github.com/mogragab/nio>.

Keywords — benchmark functions, engineering design optimization, meta-heuristic, nature-inspired algorithm, nubian ibex optimizer, optimization

1 Introduction

Optimization problems are prevalent across a wide range of fields, including engineering [52], biology [34], logistics [39], healthcare [90], artificial intelligence [11], transportation [73] and finance [79]. Recently, nature-inspired meta-heuristic algorithms have garnered considerable interest for their efficient resolution of complex optimization problems [1]. It is a class of algorithms that revolutionizes problem-solving across various domains by offering flexible and efficient techniques to tackle complex optimization challenges [1]. These flexible methods prioritize efficiency and scalability over guaranteeing optimal solutions, making them well-suited for real-world scenarios where exact algorithms may struggle. Various meta-heuristic methodologies offer unique pathways to optimization [1, 11, 34, 39, 52, 55, 73, 79, 90].

Each category of meta-heuristics offers distinct advantages and is suited to different types of problems and complexities [70]. Their ability to navigate complex solution spaces efficiently and effectively has made them indispensable tools in optimization across various fields, from engineering and logistics to finance and [Machine Learning \(ML\)](#). Meta-heuristic algorithms excel at avoiding local optima due to their stochastic nature [4]. This stochasticity enables them to navigate away from local solutions, comprehensively exploring the entire search space [24]. It effectively solves complex problems across domains such as machine learning, network design, routing, engineering, and manufacturing [24, 98]. In ML, they contribute to hyperparameter tuning, feature selection, and overall model optimization [53]. Furthermore, meta-heuristics are crucial in network design, routing, and optimization of engineering and manufacturing processes, ensuring efficient resource utilization and problem solving [25].

The versatility of meta-heuristics is a notable advantage, allowing their application to a broad spectrum of problem domains without being confined to specific types of optimization problems [70]. Furthermore, their capability for global search enhances the likelihood of finding optimal solutions within expansive solution spaces [86]. Additionally, the parallelization potential of many meta-heuristic algorithms facilitates faster convergence and scalability for addressing large-scale problems. However, the use of meta-heuristics comes with certain challenges. These algorithms do not guarantee optimal solutions, and the quality of obtained results may vary [44]. Sensitivity to parameter choices is another consideration, requiring careful tuning for optimal performance. [75] Some meta-heuristic algorithms may require substantial computational resources, which can restrict their use in contexts with limited resources [8].

The "*No Free Lunch*" theorem, formulated by Wolpert and Macready in 1997, has logically established that there is no universally superior meta-heuristic for all optimization problems [91]. It suggests that a specific meta-heuristic may perform well for one group of problems but not as well for a different group. Meta-heuristic algorithms commonly include a searching procedure that involves two main phases: diversification (exploration) and intensification (exploitation) [6]. In the initial stage, algorithms use random operators to explore different areas of the search space. During the final stage, the goal is to identify the best solution within this area. An efficient optimization algorithm must maintain a delicate balance between exploration and exploitation, ensuring a thorough search while preventing getting stuck in local optima.

In an effort to develop a more efficient and effective algorithm, this paper presents a novel nature-inspired meta-heuristic optimization algorithm, called the [Nubian Ibex Optimization \(NIO\)](#). The [NIO](#) algorithm’s design effectively captures the essence of the Nubian ibex’s behavior in both exploring new territories and exploiting known resources. By transitioning from a broad, exploratory approach to a focused, exploitative strategy, the algorithm mirrors the adaptive behaviors of the ibex, making it a robust tool for tackling complex optimization problems. To rigorously test the robustness and effectiveness of the [NIO](#), a set of twenty-three test functions is employed. Additionally, eight engineering design problems are utilized to further assess the [NIOs](#) efficacy in solving real-world challenges.

The remainder of this paper is organized as follows: Section 2 outlines the background of the introduced meta-heuristic optimization algorithms. Section 3 delves into the structure of the proposed [NIO](#). Section 4 assesses the [NIO](#) using various optimization problems, presents the outcomes and evaluations of benchmark functions and engineering problems, and compares them with current leading methods. The paper concludes with Section 5, which summarizes our findings and highlights possible directions for further investigation.

2 Related work

In the diverse landscape of computational problem-solving, seven distinct styles have emerged, each uniquely inspired by different aspects of the natural and scientific world. These include (i) evolutionary-based [3, 30, 80], (ii) swarm-based [17, 30, 72], (iii) Physics-based [17, 30, 80], (iv) human-based, (v) biology-based, (vi) system-based, and (vii) math-based approaches. Each style offers a novel perspective and methodology for tackling complex problems, drawing from the rich tapestry of knowledge ranging from biological evolution’s intricacies to music’s harmonious principles.

Evolutionary-based algorithms mimic natural evolutionary processes [18], starting with a solution pool that undergoes selection, mutation, and crossover [27] to converge toward optimal solutions [66]. Notable examples include [Genetic Algorithm \(GA\)](#) [43], which enhances candidate solutions through generations [70] and [Forest Optimization Algorithm \(FOA\)](#) [38], which simulates forest growth patterns and natural selection. Other nature-inspired approaches include [Jaguar Algorithm \(JA\)](#) [22], modeling jaguar predatory tactics to balance local and global exploration; and [Ions Motion Optimization \(IMO\)](#) [48], inspired by electrostatic forces between ions. [Lion Optimization Algorithm \(LOA\)](#) [93] models lions’ cooperative and territorial behaviors, while [Mouth Breeding Fish Algorithm \(MBF\)](#) [45] mimics mouthbrooding fish, alternating between exploration and exploitation. More recent algorithms include [Fire Hawk Optimizer \(FHO\)](#) [16], inspired by Australian kites’ hunting behavior; [Rain Optimization Algorithm \(ROA\)](#) [63], which emulates raindrops descending landscapes; and [Crested Porcupine Optimizer \(CPO\)](#) [2], which mimics porcupines’ foraging and defensive behaviors to explore solution spaces.

Swarm-based algorithms derive from collective behaviors of birds, fish, and insects, creating communal intelligence to solve complex problems [72, 95]. These algorithms adjust individual behaviors based on neighboring successes to optimize solutions. Notable examples include [Particle Swarm Optimization \(PSO\)](#) [54], [Ant Colony Optimization \(ACO\)](#) [33], [Artificial Bee Colony \(ABC\)](#) [50], [Firefly Algorithm \(FA\)](#) [17, 30, 92], [Tabu Search \(TS\)](#) [40], and [Social Spider Optimization \(SSO\)](#) [28, 47]. Each algorithm imitates specific natural behaviors. [PSO](#) simulates bird flocking or fish schooling, [ACO](#) replicates ant foraging via pheromone trails, and [ABC](#) models honeybee foraging strategies. While [TS](#) incorporates memory structures similar to human problem-solving [84], [Grey Wolf Optimizer \(GWO\)](#) [62] emulates grey wolves' leadership hierarchy and hunting mechanisms. [85]. Recent additions include [Osprey Optimization Algorithm \(OOA\)](#) [31], and [Walrus Optimizer \(WO\)](#) [42], each drawing inspiration from unique animal behaviors to balance exploration and exploitation in optimization problems.

Physics-based metaheuristic algorithms draw inspiration from physical laws, concepts, forces, and phenomena to balance exploration and exploitation in complex optimization problems [80]. These algorithms leverage analogies from particle movement, thermodynamics, fluid dynamics, and electromagnetic interactions to simulate nature's path to equilibrium or optimal states, avoiding local optima and promoting global convergence. Notable examples include [Gravitational Search Algorithm \(GSA\)](#) [71], which utilizes gravitational forces; [Water Cycle Algorithm \(WCA\)](#) [36], inspired by the water cycle; [Ray Optimization](#) [51], based on light ray behavior; and [Simulated Annealing](#) [65], derived from metallurgical annealing processes. Recent additions include [Chernobyl Disaster Optimization \(CDO\)](#) [77], inspired by the Chornobyl nuclear accident; [Energy Valley Optimization \(EVO\)](#) [17], based on energy landscapes; [Physical Phenomenon of RIME-ice \(RIME\)](#) [81], modeling ice formation; and [Prism Refraction Search \(PRS\)](#) [56], inspired by light refraction through prisms.

Human-based optimization algorithms draw inspiration from human behaviors and social dynamics to address complex computational problems [15]. These algorithms leverage the flexibility and adaptability inherent in human decision-making processes, delivering robust solutions across diverse applications [68]. They incorporate strategies inspired by queuing systems, collaborative learning, strategic games, and developmental processes to bridge human-inspired methodologies with computational problem-solving [69]. Notable examples include [Queuing Search Algorithm \(QS\)](#) [94], which models queue-waiting behaviors; [Brain Storm Optimization \(BSO\)](#) [78], inspired by brainstorming processes; and [Soccer Game Optimization \(SGO\)](#) [67], which mimics soccer game dynamics and player movements. Group dynamics inspire algorithms such as [Germinal Center Optimization \(GCO\)](#) [35], which utilizes collaborative problem-solving methods from group therapy, and [Search And Rescue Optimization \(SAR\)](#) [74], which simulates search and rescue mission prioritization strategies.

Biology-based computing draws inspiration from complex biological mechanisms to create effective computational solutions [46]. These algorithms mimic natural processes ranging from neural networks to genetic transfer and immune system reactions, demonstrating particular prominence in genetic algorithms and artificial life [29]. The inherent diversity, adaptability, and resilience of biological systems make these approaches highly effective across various applications [88]. A notable example is [Symbiotic Organisms Search \(SOS\)](#) [23], which simulates symbiotic organism interactions

through mutualism, commensalism, and parasitism phases to iteratively improve candidate solutions. Animal behavior-inspired algorithms include [Barnacles Mating Optimizer \(BMO\)](#) [13], based on avian mating behaviors; [Wild Goats Algorithm \(WGA\)](#) [76], mimicking wild goats' climbing and cooperative behaviors; [Shark Smell Algorithm \(SSO\)](#) [64], emulating sharks' hunting techniques; and [Seagull Optimization Algorithm \(SOA\)](#) [32], leveraging seagulls' foraging and migratory patterns.

System-based optimization algorithms harness the intricate dynamics and efficiencies of various natural and artificial systems to address complex computational challenges. Drawing inspiration from water cycles, ecological principles, biological growth patterns, and ecosystem interactions, these algorithms effectively handle complex problems through emergent properties. For instance, the [WCA](#) [36] mimics the natural water cycle, while the [Ecogeography-Based Optimization \(EBO\)](#) [97] applies ecogeography principles to ensure solution diversity. Furthermore, algorithms such as the [Invasive Tumor Growth Optimization \(ITGO\)](#) [83] model invasive tumor growth patterns, and the [Root Tree Optimization Algorithm \(RTO\)](#) [58] simulates tree root exploration. Similarly, the [Pity Beetle Algorithm \(PBA\)](#) [49] incorporates pity beetle behavior, whereas the [GCO](#) [87] draws from germinal center processes. Consequently, these system-based methods leverage inherent efficiency, adaptability, and complexity to bridge natural systems and computational algorithms, thereby offering distinctive and powerful approaches to optimization across diverse domains.

Math-based optimization algorithms leverage mathematical principles and models to guide the search for optimal solutions [9]. These algorithms effectively balance exploration and exploitation by utilizing the inherent properties of mathematical principles and structures, thus ensuring efficient convergence to optimal solutions. Moreover, these techniques employ rigorous mathematical concepts to address various problems, particularly in fields such as signal processing, numerical analysis, and optimization [60]. A notable example is the [Gradient Evolution Algorithm \(GEA\)](#) [57], which combines gradient descent and evolutionary principles to optimize solutions, thereby balancing local and global search capabilities. Similarly, the [Sine Cosine Algorithm \(SCA\)](#) [60] employs sine and cosine functions to explore and exploit the search space by generating new candidate solutions through position adjustments. Additionally, the [Fibonacci Indicator Algorithm \(FIA\)](#) [37] draws inspiration from the Fibonacci sequence, utilizing its ratios to guide the search process and leveraging the sequence's inherent properties for optimization.

To this end, evolutionary-based algorithms are effective in exploring large search spaces but can be computationally expensive and suffer from premature convergence. Swarm-based algorithms simulate collective behavior and are simple to implement, but may get trapped in local optima. Physics-based algorithms use physical principles for innovative solutions, but require extensive computational resources. Human-based algorithms leverage human decision-making and are intuitive but may not scale well to large problems. Biology-based algorithms mimic natural processes and are adaptable but can be computationally intensive. System-based algorithms utilize system dynamics and control theory, providing structured optimization but may be less flexible for unconventional problems.

3 Proposed Nubian Ibex Optimizer

This section presents the proposed optimizer and provides a detailed overview of its core components, including the inspiration and behavior of the Nubian ibex (3.1), the solution initialization process (3.2), the mathematical modeling of the algorithm (3.3), its pseudocode (3.4), computational complexity (3.5), and the experimental setup used to assess its performance (3.6).

3.1 Inspiration and behavior of Nubian Ibex

In the rugged terrains of the Middle East and Northeastern Africa, the Nubian Ibex stands as a remarkable adaptive species [21]. This wild goat (Fig. 1), distinguished by its impressive curved horns and sturdy build, masterfully navigates mountainous landscapes with extraordinary agility and balance. The species possesses specialized hooves featuring a soft, concave underside that creates a vacuum-like grip on sheer cliffs—a crucial adaptation for survival that allows access to areas nearly inaccessible to predators [20]. This exceptional climbing ability not only facilitates predator evasion but enables the ibex to reach fresh, ungrazed vegetation in otherwise barren environments.

The Nubian Ibex’s navigational capabilities represent a remarkable evolutionary adaptation, demonstrating an elegant combination of physical prowess, instinctual intelligence, and determination [41]. These attributes prove essential for survival in their harsh native habitats. The species typically organizes into herds ranging from small groups to congregations exceeding 20 individuals, particularly around water sources [41]. Social structure follows a gender-segregated pattern, with females and offspring forming nursery groups while males maintain either solitary lifestyles or establish smaller bachelor groups.



Figure 1: Nubian ibex

Their ability to thrive in such inhospitable landscapes, along with their social structures and mating rituals, makes them a fascinating subject for study in the fields of biology and ecology. Their presence in these harsh environments is a reminder of nature’s ability to adapt and flourish in even the most challenging conditions. These animals are also renowned for their remarkable agility

and precision in navigating the steep, rocky terrains of their habitat. This capability is particularly evident in their ability to make precise and powerful jumps to traverse challenging landscapes, a skill essential for survival and foraging in harsh environments.

Behaviorally, Ibexes have an innate ability to find the safest, most efficient routes through challenging terrain. The parallel drawn between the ibex's natural behavior and the challenges of optimization problems lies in the animal's ability to find optimal paths through rugged and unpredictable terrain efficiently. In the realm of optimization, this translates to an algorithm's capacity to navigate complex, multidimensional search spaces, seeking global optima amidst a landscape riddled with local optima and varying gradients.

3.2 Solutions Initialization

In metaheuristic optimization algorithms, the initialization of solutions is a critical phase that significantly influences the performance and outcome of the optimization process. The primary goal of solution initialization is to generate a diverse set of candidate solutions that can effectively explore the search space and avoid premature convergence to local optima. The NIO begins with an initialized herd function that mimics natural ibex dispersion across diverse terrain. This initialization phase, critical for optimization performance, generates diverse candidate solutions (X) as shown in Matrix (1) and Eq. (2). Each solution is randomly distributed within predefined upper and lower bounds (UB , LB) of the search space. This diversity is essential for exploring different regions simultaneously and preventing premature convergence to suboptimal solutions [59]. As the algorithm progresses through its iterations, the most effective solution obtained at any given point is provisionally identified as the optimal solution. This iterative process is essential in gradually steering the search towards the most promising regions of the solution space, thereby enhancing the efficiency and effectiveness of the NIO algorithm in identifying a solution that closely approximates the optimal result.

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,D-1} & x_{1,D} \\ x_{2,1} & \cdots & x_{2,j} & \cdots & x_{2,D-1} & x_{2,D} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,D-1} & x_{i,D} \\ x_{N-1,1} & \cdots & x_{N-1,j} & \cdots & x_{N-1,D-1} & x_{N-1,D} \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,D-1} & x_{N,D} \end{bmatrix} \quad (1)$$

where the variable X_i symbolizes a collection of current candidate solutions; these solutions are generated in a random manner using Eq. (1). Within this context, X_i represents the decision values, or referred to as the positions, of the i^{th} solution in the set. The total number of these candidate solutions is encapsulated by the variable N , which effectively represents the size of the population in the optimization problem. Additionally, the term D is used to denote the dimension size of the problem under consideration. This dimension size is a critical aspect, as it defines the complexity and the scope of the problem space that the NIO algorithm navigates during the optimization process.

$$X_{ij} = R \times (UB_j - LB_j) + LB_j; \quad (2)$$

for $i = 1, 2, \dots, N; j = 1, 2, \dots, D$

where R signifies a random variable, LB_j refers to the lower bound of the j^{th} dimension, and UB_j indicates the upper bound of the j^{th} dimension within the given problem, N is the number of solutions or entities, while D represents the number of dimensions in the problem. Initialization strategies play a key role in optimization algorithms by balancing exploration and exploitation to avoid premature convergence. These strategies span from basic random uniform distribution to advanced knowledge-based heuristics. Each method distributes agents uniquely across the search space, significantly influencing algorithm performance in complex solution landscapes.

3.3 Mathematical Model

The NIO algorithm draws inspiration from the Nubian Ibex, leveraging its remarkable traits of resilience and efficiency in challenging terrains as a metaphorical framework for optimization. This approach introduces a population of "ibexes" that represent potential solutions, navigating the search space through iterative updates guided by both individual and global best solutions.

The algorithm specifically incorporates the ibex's jumping strength, agility, and precision to enhance performance in complex optimization problems. A distinctive feature of NIO is its memory mechanism, where ibexes remember historically promising areas, which influences their movement patterns and velocity updates, thereby improving the balance between exploration and exploitation. The algorithm addresses scalability considerations, maintaining effectiveness even with large-scale optimization problems by optimizing computational steps and efficiently handling high-dimensional and noisy landscapes.

The primary motivation for NIO is to create a novel metaheuristic algorithm that effectively balances discovering new promising areas with intensively searching within those areas for optimal solutions. Fig. 2 illustrates the moving geometry of the Nubian ibex, depicting the conceptual framework of the algorithm as it navigates the complex and rugged landscape of the solution space.

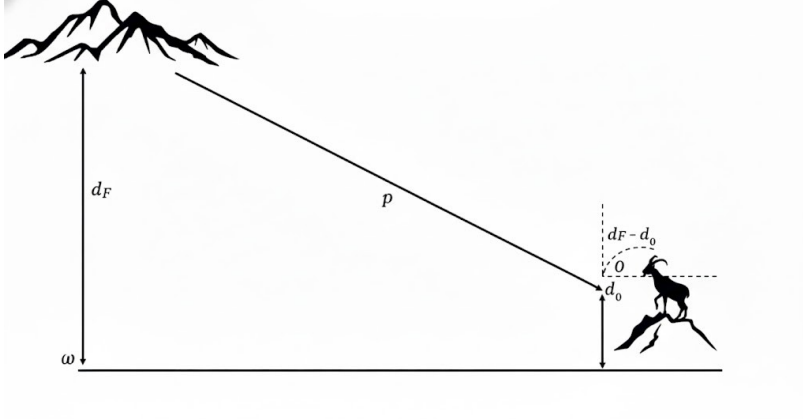


Figure 2: Moving geometry of Nubian Ibex. The angle θ denotes the movement trajectory angle (or trajectory adjustment angle) calculated from the elevation difference ($d_F - d_o$) and the path curvature (ρ). This visualization illustrates how ibexes adjust their jumping trajectory on rugged terrain, where d_o is the initial elevation, d_F is the final elevation, and ω is the horizontal distance to the target position.

The ibex's path adjustment mechanism is modeled through a movement angle θ that reconciles vertical elevation gain with horizontal displacement. While direct biomechanical data validating the exact jump geometry of the Nubian ibex are not available in the zoological literature, our Eq. (3) is a mathematical abstraction based on behavioral observation and common modeling principles in animal-inspired optimization. This approach follows established practices in recent metaheuristic algorithm research, where geometric, angle-based, and trajectory-inspired movement models are regularly employed without direct biological parameterization. Notable examples include the [Aquila Optimizer \(AO\)](#) [7] (2021), Harris [Harris Hawks Optimization \(HHO\)](#) [12] (2021), and the [Walrus Optimization Algorithm \(WAOA\)](#) [42] (2024). The geometry, illustrated in Figure 2, computes the optimal launch angle for traversing between positions while accounting for landscape curvature ρ :

$$\theta = \sin^{-1} \left(\frac{d_F - d_o}{\rho} \right) = \tan^{-1} \left(\frac{d_F - d_o}{\omega} \right) \quad (3)$$

where θ is the movement angle, d_F is the distance to the final position, d_o is the distance from the origin, ρ is the path's radius of curvature, and ω is the distance to the point of interest.

The scale factor σ is determined using the gamma function Γ and a constant β (value 1.5), as shown in Eq. (4):

$$\sigma = \left(\frac{\Gamma(1 + \beta) \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \quad (4)$$

The jump strength, denoted as ψ , is a critical parameter in the NIO algorithm, influencing how solutions are updated during the optimization process. It is calculated using Eq. (5):

$$\psi = \frac{R_{norm}(D) \cdot \sigma}{|R_{norm}(D)|^{\frac{1}{\beta}}} \quad (5)$$

where $R_{norm}(D)$ represents a vector containing random numbers that follow a normal distribution, D denotes the dimension of this vector, and σ is a scaling factor. This equation adjusts each element of $R_{norm}(D)$ by a scale factor influenced by β that controls the influence of the random component on the jump strength. This formulation introduces stochasticity in the jump strength, allowing the algorithm to explore the search space effectively.

By mimicking natural behaviors, the NIO algorithm effectively explores new solutions while exploiting known good ones, making it a robust and versatile tool for solving complex optimization problems. Balancing these two processes is crucial for the success of any optimization algorithm. The mechanisms by which the optimization process achieves this balance, specifically through the mean position (X_{mean}) and the global best position (X_{best}) of the ibexes in the herd. NIO improves optimization dynamics unlike WGA that typically updates each solution (goat) based on leader-following and random exploration.

The mean position, X_{mean} , serves as a central reference point for the herd of ibexes, providing insights into their collective movement and guiding individual members toward optimal solutions. By indicating the overall trend of the herd's movement, X_{mean} helps balance exploration and exploitation, allowing the herd to fine-tune its adjustments and converge more effectively. It acts as a collective indicator, reflecting the general direction of the herd and helping individual ibexes make informed decisions about their trajectories, ensuring a more efficient search process.

Moreover, X_{mean} plays a crucial role in guiding the herd toward convergence by reducing the variance in positions over iterations. This reduction in variance leads the herd to cluster around potential optimal solutions, enhancing the efficiency and effectiveness of the optimization process. By maintaining a balance between exploration and exploitation, X_{mean} ensures that the search remains focused on promising areas while still allowing for sufficient exploration to avoid premature convergence to suboptimal solutions.

The position X_{best} represents the current best-known solution within the herd, guiding the search towards promising areas in the search space. This helps in directing the movement of the herd towards this optimal solution, enhancing search efficiency. Additionally, X_{best} serves as a reference point, allowing individual ibexes to adjust their trajectories and move closer to the optimal solution, improving the likelihood of converging to the global optimum. Lastly, X_{best} acts as a historical best reference, retaining the best solution found through iterations. This prevents the herd from getting stuck in local optima and ensures the search remains focused on the best solutions, enhancing the robustness of the optimization process.

While WGA relies on simple leader-following and random movement mechanisms, NIO integrates biologically inspired behaviors that enable more adaptive and robust optimization performance. NIO provides a more dynamic and biologically realistic approach for solving optimization problems. This mechanism helps the algorithm recover from local optima and reuse good knowledge, which

WGA lacks entirely. It allows better balance between exploration and exploitation, greater solution diversity, and superior convergence accuracy.

The mathematical framework of NIO comprises three key phases: (i) exploration phase (3.3.1), which implements strategies for diverse searching across the solution space; (ii) exploitation phase (3.3.2), which focuses on refining known good solutions; and (iii) balancing mechanism (3.3.3) that dynamically adjusts between exploration and exploitation, in addition to the adaptive boundaries clipping (3.3.4).

3.3.1 Exploration Phase

The exploration phase of the NIO algorithm focuses on comprehensive scanning of the search space to identify potential areas of interest. This phase, occurring during the first two-thirds of iterations, employs an aggressive search strategy inspired by ibexes exploring new terrains. The algorithm utilizes a combination of the best position discovered thus far and the ibex jump function to explore new areas, as represented in Eq. (6):

$$X_i^{new} = X_{best} + \psi(D) + X_{rand} \cdot R \quad (6)$$

The algorithm also incorporates an average position update method defined in Eq. (7), which calculates new positions using a weighted average approach:

$$X_i^{new} = X_{best} \cdot \left(1 - \frac{N}{T}\right) + (X_{mean} - X_{best}) \cdot \theta \quad (7)$$

To maintain effective exploration, the algorithm leverages dynamic strategies that incorporate both global best positions and individual search paths. The ibex jump function allows solutions to leap to new areas, preventing confinement to the vicinity of current best solutions. Diversity is preserved through random initialization and stochastic jumping mechanisms, enhancing the probability of discovering global optima. The strength and direction of jumps adapt based on iteration number and relative performance, ensuring flexibility in exploring both distant and near regions of the search space.

The exploration behavior is further enhanced through the incorporation of social dynamics observed in actual ibex herds, as illustrated in Fig. 3. Individual ibex movement patterns are influenced by the collective behavior of the herd, creating a balanced exploration strategy that prevents premature clustering around suboptimal solutions.

3.3.2 Exploitation Phase

In the exploitation phase, occurring during the final third of iterations, the NIO algorithm transitions from broad search to concentrated focus on promising areas identified during exploration. This phase is essential for fine-tuning solutions and achieving optimal results. The algorithm's parameters of agility and precision become increasingly significant, allowing swift adjustment of positions and more accurate refinements.



Figure 3: Nubian ibex exploration

The transition to exploitation parallels the behavioral adaptation observed in Nubian ibex herds when they approach specific objectives within their environment. As illustrated in Fig. 4, the animals exhibit increasingly deliberate and targeted movement patterns when navigating toward essential resources or traversing particularly challenging terrain segments. This biological inspiration directly informs the algorithmic implementation, where the computational search process becomes progressively more focused and methodical.

During this phase, the algorithm employs two key position update methods: the agility and precision-based update defined in Eq. (8):

$$X_i^{new} = (X_{best} - X_{mean}) \cdot \alpha - R + ((UB - LB) \cdot R + LB) \cdot \vartheta \quad (8)$$

And the jump and agility-based update defined in Eq. (9):

$$X_i^{new} = X_{best} - (\tilde{a} \cdot X_i \cdot R) - \tilde{\theta} \cdot \tau(D) + R \cdot \tilde{a} \quad (9)$$

The algorithm concentrates search around the best solutions with shorter, more precise jumps. Even during this exploitation-focused phase, the algorithm maintains a balance by allowing occasional broader jumps, ensuring comprehensive coverage of the search space and preventing entrapment in local optima.

3.3.3 Balancing Exploration and Exploitation

Achieving an optimal balance between exploration and exploitation is crucial for optimization algorithm effectiveness. The NIO algorithm dynamically adapts its search strategies based on the current optimization phase to maintain this balance. Initially, the algorithm emphasizes exploration for comprehensive search, gradually transitioning to exploitation as the process advances. The algorithm employs a feedback mechanism where current solution performance influences search strategy, potentially reverting to more exploratory approaches during stagnation to escape local optima.

The update process consists of several key stages: (i) bounds check, which constrains each dimension of X_i within specified limits; (ii) fitness evaluation, which assesses each solution's fitness



Figure 4: Nubian ibex exploitation movements

and updates the global best position X_{best} when superior solutions are found; and (iii) parameter adaptation, which adjusts the agility (α) and precision (ϑ) parameters as iterations progress.

For solutions in group X_2 , the algorithm selects random solutions from the elite group X_1 , moving X_i towards the selected elite solution X_j . The parameters α and ϑ control the algorithm's behavior, with larger values leading to faster convergence but potential overshooting, while smaller values result in more conservative updates with slower convergence but finer-grained optimization.

3.3.4 Adaptive Boundaries Clipping

Maintaining solution candidates within specified bounds is fundamental for optimization algorithms to ensure search process feasibility and efficiency. The NIO algorithm implements a clipping mechanism that enforces these bounds, enhancing the algorithm's ability to focus on promising search space regions. The clipping function is defined in Eq. (10):

$$Clip(P_i) = \begin{cases} LB & \text{if } P_i < LB \\ P_i & \text{if } LB \leq P_i \leq UB \\ UB & \text{if } P_i > UB \end{cases} \quad (10)$$

where P_i is an individual population element. This mechanism is vital for: (i) dynamic adaptation to the feasible search region, (ii) enhancing algorithmic agility through quick boundary adjustments, and (iii) increasing precision by ensuring all candidate solutions are valid. The updating of the best solution S^* and its corresponding fitness F^* is described by Eq. (11):

$$S^* \leftarrow P[I^*], F^* \leftarrow F[I^*] \quad (11)$$

where P is the population, I^* is the index of the best individual, and F contains fitness values. The text also notes that NIO improves upon WGA by employing strategies like delayed regrouping and territorial separation to avoid premature convergence and maintain diversity, alongside memory-based elements inspired by ibex behavior, making it more adaptable for complex optimization problems.

3.4 NIO Pseudocode

In the NIO algorithm, the optimization process begins by creating an initial diverse group of potential solutions, referred to as the population. As the algorithm iterates, the search mechanisms emulate the ibex's movement patterns to efficiently scout the vicinity of the most promising solutions or the best solution identified. Every potential solution is dynamically adjusted to the best solution found during the optimization steps of the algorithm.

To ensure a balance in the NIO algorithm's search approach (that is, exploration and exploitation), four distinct search methods are implemented: extensive exploration, precise exploration, extensive exploitation, and precise exploitation. The optimization cycle concludes once a predefined termination condition is satisfied. The step-by-step procedure of the NIO algorithm is elaborated in Algorithm 1.

Algorithm 1: Nubian Ibex Optimization (NIO)

Data: Population size N , Dimension D , Boundaries $[LB, UB]$, Iterations T
Result: Best solution S^* and its fitness value F^*

- 1 Initialize population P with random values in the range $[LB, UB]$ using (2)
- 2 **For** $t = 1 \rightarrow T$ **Do**
- 3 **Foreach** individual X_i in P **Do**
- 4 Evaluate fitness F_i of X_i using (11)
- 5 Find the index I^* of the individual with the best fitness
- 6 $I^* \leftarrow$ index of the individual with the best fitness in P
- 7 $S^* \leftarrow P[I^*]$
- 8 $F^* \leftarrow F[I^*]$
- 9 **For** $i = 1 \rightarrow N$ **Do**
- 10 **If** $i \neq I^*$ **Then**
- 11 Calculate angle θ using (4)
- 12 Update position of individual $X[i]$ using (6) — (9)
- 13 Clip $P[i]$ to be within the range $[L, U]$ using (10)
- 14 **Return** S^*, F^*

3.5 Computational Complexity

The computational complexity of the NIO algorithm is primarily determined by three core operations: the initialization of solutions, the evaluation of fitness functions, and the iterative process of updating these solutions. During the initialization phase, the complexity is linearly proportional to the number of ibexes, denoted as N , resulting in a complexity of $O(N)$. The subsequent updating mechanism within the NIO algorithm exhibits a more intricate complexity, which is characterized by the sum of two terms: $O(T \times N)$ and $O(T \times N \times D)$. Here, $O(T \times N)$ represents the complexity

associated with searching for the best location, while $O(T \times N \times D)$ accounts for updating the location vector for all ibexes, where T signifies the maximum number of iterations and D indicates the dimensionality of the problem.

Consequently, the overall run time complexity, encompassing both initialization and the iterative updates, is formulated as $O(N \times (T + T \times D + 1))$. This expression underscores that the algorithm’s runtime scales linearly with the number of ibexes (N) and demonstrates a more complex dependency on the product of the number of iterations (T) and the problem’s dimensions (D), in addition to a linear term related to the number of ibexes. Therefore, the efficiency and scalability of the NIOs are significantly influenced by these parameters, showing a linear increase in complexity with N and a more nuanced rise linked to the interplay between T and D .

3.6 Experimental Setup

This section outlines the complete experimental setup used to evaluate the performance of the NIO algorithm. We outline the implementation tools, including the programming languages and libraries employed, to facilitate replication and further investigation by other researchers. This section includes the following components: the computational environment and system requirements (3.6.1), the internal parameter configuration of NIO (3.6.2), the evaluation metrics used to measure algorithm performance (3.6.3), the benchmark metaheuristics algorithms (3.6.4), and the suite of benchmark functions and real-world problems employed in testing (3.6.5).

3.6.1 System Requirements

Experiments were executed on a Windows 11 system with Intel Xeon 2.20 GHz CPU and 64 GB RAM. NIO’s parameterization was carefully tuned to balance exploration and exploitation. Key settings include a discovery rate ($\lambda = 0.2$), jumping rate ($\gamma = 0.3$), and radius rate ($\rho = 0.5$). Adaptive mechanisms govern agility ($\alpha = 0.1$) and precision ($\vartheta = 0.05$), promoting dynamic optimization. Initial solutions are sampled via uniform distribution, with stochastic variation maintained using uniform randomization over $[0, 1]$. These configurations collectively ensure that NIO is both computationally efficient and effective across diverse problem landscapes, demonstrating promising convergence behavior and competitive results compared to state-of-the-art algorithms. The detailed requirements are shown in Table 1.

Table 1: Hardware and software configuration

Parameter	Value
Operating System	Windows 11 system with Intel Xeon 2.20 GHz
RAM	64 GB DDR4
Framework	PyTorch 2.0
Compilers/IDE	Anaconda 2024.03 (VS Code)
Python	Version 3.12

3.6.2 Algorithm Parameters

The performance and convergence behavior of the NIO algorithm are significantly influenced by its internal parameters. Table 2 summarizes the primary parameters used in our experiments, along with their descriptions and the selected values based on preliminary testing and standard practices in optimization algorithms.

The population size of 50 maintains solution diversity with reasonable computational cost, while 1200 iterations allow sufficient time for convergence. Agility (1.2) controls the agents' responsiveness to the landscape, enhancing adaptive movement, whereas precision (0.01) enables fine-tuning near optima. A jumping rate of 0.3 allows occasional long-range moves to escape local traps, and randomization in the range $[0, 1]$ introduces controlled stochasticity to prevent premature convergence and maintain search diversity.

Table 2: Default parameter settings

Parameter	Value
Population (N)	[50 : 600]
Iterations (T)	[100 : 1200]
Discovery Rate (λ)	0.2
Jumping Rate (γ)	0.3
Radius Rate (ρ)	0.5
Agility (α)	Addaptive — 0.1
Precision (ϑ)	Addaptive — 0.05
Initialization (I)	Random Uniform
Randomization (R)	Uniform in $[0, 1]$

3.6.3 Evaluation Metrics

The evaluation of optimization algorithms, including the proposed NIO algorithm, requires comprehensive statistical metrics to evaluate performance between multiple independent runs on benchmark functions. The study employs five key metrics, including best, worst, mean, standard deviation, and median fitness, to provide a multifaceted evaluation of the effectiveness of the algorithm.

The *Best fitness*, Eq. (12), value represents the minimum value achieved between runs, indicating the exploitation capacity of the algorithm. In contrast, the *Worst fitness*, Eq. (13), value highlights the maximum value obtained, revealing the algorithm's worst-case performance and robustness. The *Mean fitness*, Eq. (14), value calculates the average performance across all runs, serving as a measure of overall effectiveness and reliability. The *Standard Deviation*, Eq. (15), quantifies performance consistency, with lower values indicating more stable behavior. Finally, the *Median fitness*, Eq. (16), value provides the middle value of all runs, offering a robust measure that is less sensitive to outliers than the mean.

$$Best = \min_{1 \leq i \leq K} f_i, \quad (12)$$

$$Worst = \max_{1 \leq i \leq K} f_i, \quad (13)$$

$$Mean = \frac{1}{K} \sum_{i=1}^K f_i, \quad (14)$$

$$Std = \sqrt{\frac{1}{K} \sum_{i=1}^K (f_i - Mean)^2}, \quad (15)$$

$$Median = \begin{cases} f_{(K+1)/2} & \text{if } K \text{ is odd,} \\ \frac{f_{K/2} + f_{(K/2)+1}}{2} & \text{if } K \text{ is even,} \end{cases} \quad (16)$$

In all the above equations, f_i denotes the fitness value obtained in the i -th independent run, and K represents the total number of runs performed. The term *Mean* refers to the average of all fitness values and is used in the calculation of the standard deviation and other statistical measures.

3.6.4 Benchmark Metaheuristics Algorithms

Comparative analysis is conducted against established metaheuristics including SCA [60] (2016), Whale Optimization Algorithm (WOA) [61] (2016), Queuing Search Algorithm (QSA) [94] (2018), Nuclear Reaction Optimization (NRO) [89] (2019), Gradient-Based Optimizer (GBO) [10] (2020), Artificial Ecosystem Optimization (AEO) [96] (2020), BMO [82] (2020), Golden Jackal Optimization (GJO) [26] (2022), and CDO [77] (2023).

All the evaluated algorithms used the same set of benchmark functions and the mean rank test was used to illustrate NIOs performance. For all methods, the number of iterations and population size were set to 200 and 50, respectively, conducted over 50 independent runs for both the test benchmarks and engineering problems.

3.6.5 Benchmark Functions and Problems

The proposed NIO was evaluated using two complementary test suites: a set of 23 benchmark functions and 8 real-world engineering design problems. The benchmark functions cover three categories—unimodal, multimodal, and fixed-dimension multimodal—enabling a thorough assessment of NIO’s convergence behavior, exploration–exploitation balance, and scalability.

Unimodal functions (F_1 – F_7) assess NIO’s exploitation ability, focusing on precision and convergence in smooth search landscapes. *Multimodal functions* (F_8 – F_{13}) evaluate exploratory capacity, testing the algorithm’s ability to avoid local optima and maintain population diversity. *Fixed-dimension multimodal functions* (F_{14} – F_{23}) gauge scalability and performance sensitivity under dimensional constraints. Full details, including function definitions, domains, and global optima, are summarized in Table 3.

To further validate its practical effectiveness, NIO is also applied to a diverse set of engineering problems, including the Tension/Compression Spring Design Problem, Pressure Vessel Design Problem, Speed Reducer Problem, Three Bar Truss Problem, Welded Beam Design Problem, Cantilever Beam Design Problem, Multiple-disc Clutch Brake Problem, and Piston Lever Design Problem. This combined evaluation framework highlights the algorithm’s robustness across both theoretical and real-world optimization scenarios.

4 Results and Discussion

The performance of the NIO algorithm was evaluated across 23 benchmark functions using a suite of visual and quantitative tools. Visualization techniques—such as surface and contour plots, 2D/3D trajectory maps, and exploration-exploitation graphs—were employed to characterize the search dynamics throughout the optimization process (Fig. 5). Surface plots depict the fitness landscape, while contour maps provide 2D equi-functional insights. The exploration-exploitation graph quantifies NIO’s ability to balance global search with local refinement, affirming its robust and adaptive behavior. The rest of this analysis is structured into the following subsections: exploitation on unimodal functions (4.1), exploration behavior in multimodal landscapes (4.2), robustness on fixed-dimension functions (4.3), adaptive dynamics of the algorithm (4.4), parameter analysis (4.5), and run-time complexity (4.6).

4.1 Unimodal Functions: Exploitation Performance

Across the benchmark suite and as shown in Table 4 and Fig. 6, NIO consistently outperformed existing metaheuristics, driven by its biologically inspired angle-based exploitation mechanism, which mimics the precise, terrain-aware movements of the Nubian ibex. This adaptive navigation enables efficient convergence while avoiding premature stagnation.

In particular, NIO exhibited dominant performance on unimodal functions, especially F_1 through F_4 . On F_1 , a shifted variant of the Sphere function, NIO achieved a best value of 7.334×10^{-172} —nearly 100 orders of magnitude better than the closest competitor (AEO at 1.73×10^{-79}). Moreover, NIO’s mean and worst results (1.454×10^{-144} and 7.271×10^{-143} , respectively) surpassed even the best outcomes of competing algorithms. This trend continued across F_2 to F_4 , underscoring NIO’s consistency, precision, and superior convergence.

The core strength of the NIO algorithm lies in its angle-based exploitation mechanism, enabling fine-grained local search through incremental orientation adjustments. Inspired by the Nubian ibex’s terrain-aware navigation, this strategy avoids overshooting and enhances stability—particularly beneficial for unimodal functions where precision is critical. As the landscape complexity increases (e.g., F_5), performance margins narrow, yet NIO maintains superiority with a best result of 5.247×10^{-6} . Its success is attributed to the interplay between population diversity and adaptive step refinement, allowing the algorithm to effectively explore deceptive plateaus and uncover deep minima.

Table 3: Benchmark functions.

F	Range	Dim
$F_1(x) = \sum_{i=1}^n x_i^2$	$[-100, 100]$	30
$F_2(x) = \sum_{i=0}^n x_i + \prod_{i=0}^n x_i $	$[-10, 10]$	30
$F_3(x) = \sum_{i=1}^d (\sum_{j=1}^i x_j)^2$	$[-100, 100]$	30
$F_4(x) = \max\{ x_i , 1 \leq i \leq n\}$	$[-100, 100]$	30
$F_5(x) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	$[-30, 30]$	30
$F_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	$[-100, 100]$	30
$F_7(x) = \sum_{i=0}^n ix_i^4 + \text{random}[0, 1]$	$[-1.28, 1.28]$	30
$F_8(x) = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$	$[-500, 500]$	30
$F_9(x) = -(x_2 + 47) \sin\left(\sqrt{ x_2 + \frac{x_1}{2} + 47 }\right) - x_1 \sin\left(\sqrt{ x_1 - x_2 - 47 }\right)$	$[-5.12, 5.12]$	30
$F_{10}(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	$[-32, 32]$	30
$F_{11}(x) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]$	30
$F_{12}(x) = -20e^{(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2})} - e^{(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i))} + 20 + e$	$[-50, 50]$	30
$F_{13}(x) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-50, 50]$	30
$F_{14}(x) = \sqrt[8]{(\ x\ ^2 - n)^2} + \frac{1}{n} \left(\frac{1}{2}\ x\ ^2 + \sum_{i=1}^n x_i\right) + \frac{1}{2}$	$[-65, 65]$	2
$F_{15}(x) = -\sum_{i=1}^n \sin(x_i) \left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{20}$	$[-5, 5]$	4
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	2
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right) \cos x_1 + 10$	$[-5, 15]$	2
$F_{18}(x) = 1 + \sum_{i=1}^n \sin^2(x_i) - 0.1e^{(\sum_{i=1}^n x_i^2)}$	$[-2, 2]$	2
$F_{19}(x) = -\sum_{i=1}^4 c_i \exp\left(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]$	3
$F_{20}(x) = -\sum_{i=1}^4 c_i \exp\left(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]$	6
$F_{21}(x) = -\sum_{i=1}^5 \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4
$F_{22}(x) = -\sum_{i=1}^7 \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4
$F_{23}(x) = -\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]$	4

The exploitation analysis conducted using unimodal benchmark functions reveals NIO’s exceptional performance in fine-tuning solutions near optimal regions. The algorithm demonstrates highly effective results on functions F_1 , F_2 , F_3 , and F_7 , surpassing all competing algorithms and confirming its excellence in extracting optimal solutions within unimodal landscapes. This superior exploitation capability aligns with the algorithm’s ability to intensify searches in promising regions while maintaining precise solution refinement.

4.2 Multimodal Functions: Exploration Behavior

In functions F_6 through F_8 , NIO reestablishes dominance—most notably on F_8 , where it achieves best, mean, and worst values of -1.257×10^4 , -1.256×10^4 , and -1.254×10^4 , respectively. This consistency stems from the collective behavior of ibex-inspired agents, which collaboratively adjust their orientation to converge on optimal basins.

NIO also demonstrates exceptional precision in zero-optimum functions such as F_9 , F_{10} , and F_{11} , achieving exact global minima ($0.e + 00$) across all metrics. This highlights its intensification capability, where subtle angular refinements ensure convergence without oscillation, and its ability to transition smoothly from exploration to exploitation.

For complex multimodal landscapes (e.g., F_{12} and F_{13}), NIO achieves competitive results (5.389×10^{-9} and 2.613×10^{-7}), benefiting from dynamic step reorientation that allows responsive adaptation across shifting functional regions.

The exploration analysis utilizing multimodal functions with multiple local optima demonstrates NIO’s strong performance in navigating complex landscapes. The algorithm achieves superior mean values in functions F_{10} , F_{12} , and F_{13} , successfully identifying global optima for functions F_{19} , F_{21} , and F_{23} . These results highlight the algorithm’s capability to maintain population diversity and effectively explore various search space regions while avoiding local optima traps.

4.3 Fixed-Dimension Multimodal: Robustness and Scalability

While the NRO algorithm records a better mean on some functions, NIO excels in the benchmarks of F_{14} , F_{19} , and F_{21} , outperforming all other algorithms. Notably, NIO successfully identifies the global optimum for functions F_{19} , F_{21} , and F_{23} , demonstrating its proficiency in exploration and its ability to avoid local optima traps.

These results highlight NIOs capability to maintain diversity within the population and effectively explore various regions of the search space. By preventing premature convergence and encouraging exploration, NIO increases the likelihood of discovering the global optimum in complex multimodal landscapes. Fig. 5 illustrates the qualitative results of NIO on the benchmark functions, showcasing its search trajectory and convergence patterns.

4.4 Adaptive Exploration Exploitation Balance

While NIO excels in a majority of cases, performance dips are observed in certain functions such as F_{15} and F_{20} , where alternative algorithms marginally outperform it. These instances reaffirm the

No Free Lunch Theorem, underscoring that no single method is optimal for all problems. Nonetheless, NIO remains consistently competitive across metrics. A key contributor to this performance is NIO’s adaptive exploration–exploitation balance, driven by stochastic jump mechanisms and diversity preservation in early phases, which ensure broad search space coverage. As iterations progress, angular adjustments and adaptive step control guide the algorithm toward stable, high-precision convergence.

To quantitatively substantiate this behavior, an exploration–exploitation analysis was conducted by tracking normalized population diversity over iterations. Results reveal a gradual, adaptive transition: on average, exploration decreases from 88.40% at initialization to 45.57% at the midpoint and 15.29% in the final half, settling at 11.11% by termination, with the mean transition point occurring near iteration 40.7. Furthermore, this behavior adapts to function complexity, NIO transitions earlier on simple unimodal functions, sustains exploration longer on non-separable functions, and retains higher terminal diversity on multimodal landscapes, effectively reducing premature convergence.

During later stages, NIO intensifies its search through precision-adjustment features that dynamically adapt search parameters, focusing computational efforts on promising regions while preserving coverage of unexplored areas, ensuring both solution diversity and convergence quality.

To sum up, the comprehensive evaluation establishes NIO as a significant advancement in metaheuristic optimization, validated through both benchmark performance and empirical exploration–exploitation analysis. Its biomimetic, angle-based search mechanism provides fine-grained control over solution updates, enabling consistent and competitive performance across diverse optimization landscapes, making it a valuable and practical tool for complex optimization problems.

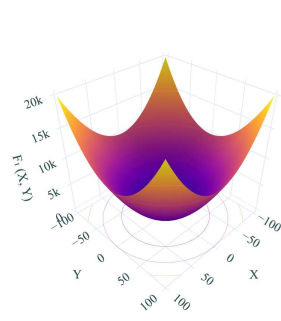
4.5 Parameter Analysis

We assessed the sensitivity of the NIO algorithm’s performance to three crucial parameters: iteration count (T), population size (N), and initialization method (I). These parameters significantly impact the balance between exploration and exploitation during optimization.

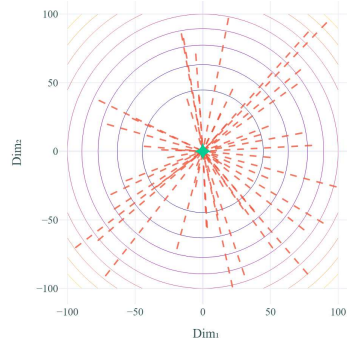
Increasing iterations from 50 to 1200 showed varied improvements across benchmark functions (Table 5). For instance, function F_1 reached optimal convergence at $T = 400$, whereas F_{21} and F_{23} attained optimal solutions early, benefiting minimally from additional iterations. Conversely, functions such as F_5 and F_8 required higher iterations for substantial performance enhancement.

Population size evaluation (range: 25–600) revealed diverse function-specific behaviors (Table 6 and Fig. 7). Optimal results for F_5 were observed at the highest population size ($N = 600$, 8.543×10^{-5}), while F_1 favored moderate population sizes ($N = 50$). Functions F_8 and F_{21} exhibited robustness across varying population sizes.

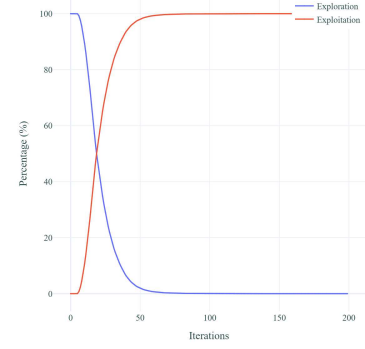
Examining ten initialization strategies (Table 7) highlighted function-specific optimal choices: Centroid initialization excelled for F_1 , whereas Random initialization was optimal for F_{12} and F_{23} . Multiple initializers produced equally strong outcomes for F_8 and F_{16} , indicating NIO’s initialization adaptability.



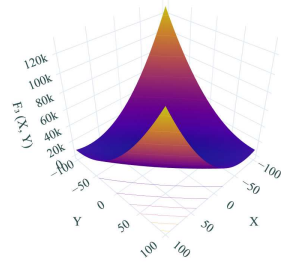
F₁ Surface



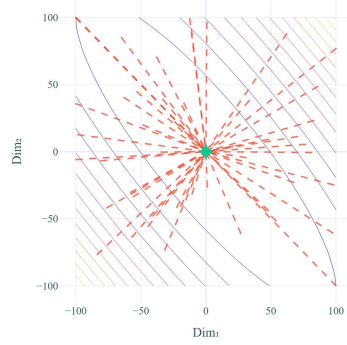
F₁ Contour Path



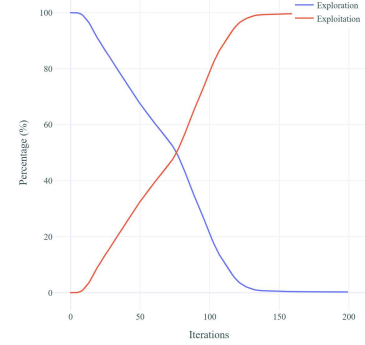
F₁ Search Dynamics



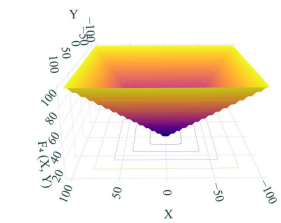
F₃ Surface



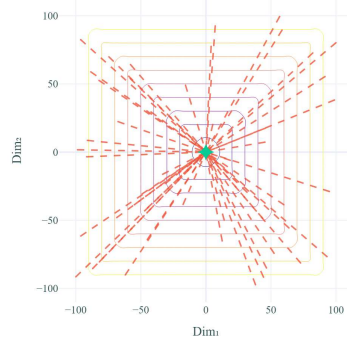
F₃ Contour Path



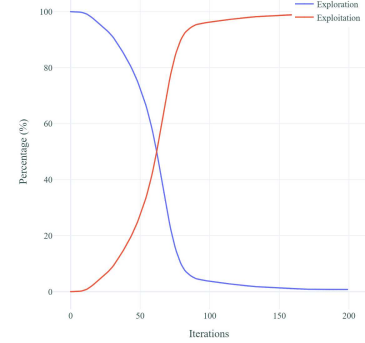
F₃ Search Dynamics



F₄ Surface



F₄ Contour Path



F₄ Search Dynamics

Figure 5: Qualitative results for the studied problems.

Table 4: Results of benchmark functions.

F	Measure	Algorithms									
		SCA	WOA	QSA	NRO	GBO	AEO	BMO	GJO	CDO	NIO
F ₁	Best	8.85e-02	1.46e-20	2.29e+02	1.02e-07	6.44e-12	1.73e-79	9.13e-88	4.29e-11	2.94e-30	7.33e-172
	Mean	4.62e+01	1.4e-14	6.84e+02	1.82e-03	8.6e-07	3.69e-70	1.39e-72	8.87e-10	7.39e-28	1.45e-144
	Worst	4.76e+02	1.24e-13	1.71e+03	1.88e-02	3.e-05	1.57e-68	3.77e-71	7.43e-09	1.56e-26	7.27e-143
F ₂	Best	6.53e-03	2.07e-13	6.86e+00	1.47e-04	1.64e-06	6.7e-42	5.85e-45	3.42e-07	4.97e-16	7.78e-87
	Mean	2.01e-01	5.06e-10	1.32e+01	1.45e-02	5.59e-04	4.7e-36	9.57e-39	1.38e-06	3.11e-15	3.07e-78
	Worst	9.65e-01	7.25e-09	2.07e+01	8.93e-02	3.48e-03	7.56e-35	1.58e-37	3.02e-06	1.44e-14	7.79e-77
F ₃	Best	2.96e+03	4.07e+04	7.35e+03	8.42e-05	1.49e-10	7.00e-81	1.49e-95	5.89e-04	2.16e-28	1.32e-164
	Mean	9.04e+03	7.23e+04	1.43e+04	2.43e-01	4.72e-05	3.06e-70	1.06e-72	1.4e-01	5.02e-25	1.85e-130
	Worst	1.70e+04	1.11e+05	3.01e+04	3.99e+00	8.09e-04	1.18e-68	4.37e-71	4.56e+00	6.45e-24	9.23e-129
F ₄	Best	1.22e+01	5.94e+01	2.5e+01	5.32e-04	8.31e-07	8.59e-42	1.5e-47	2.76e-04	4.67e-14	4.22e-85
	Mean	2.67e+01	7.91e+01	3.91e+01	4.83e-02	1.65e-04	1.21e-35	4.01e-38	2.69e-03	2.05e-13	1.66e-78
	Worst	3.85e+01	8.99e+01	5.21e+01	3.52e-01	1.04e-03	1.93e-34	8.47e-37	7.78e-03	9.47e-13	3.43e-77
F ₅	Best	8.24e+01	2.84e+01	3.20e+04	2.81e+01	2.79e+01	2.88e+01	2.89e+01	2.67e+01	2.82e+01	5.25e-06
	Mean	5.02e+04	2.87e+01	1.86e+05	2.88e+01	2.85e+01	2.89e+01	2.9e+01	2.77e+01	2.85e+01	4.99e-02
	Worst	2.45e+05	2.88e+01	9.06e+05	3.01e+01	2.88e+01	2.9e+01	2.9e+01	2.88e+01	2.87e+01	4.79e-01
F ₆	Best	5.01e+00	6.07e-01	1.23e+02	6.41e-01	4.55e-01	3.09e+00	5.69e+00	7.7e-01	7.5e+00	1.91e-08
	Mean	5.60e+01	1.34e+00	7.88e+02	1.52e+00	1.54e+00	4.85e+00	6.60e+00	1.64e+00	7.5e+00	1.56e-04
	Worst	3.66e+02	2.28e+00	2.27e+03	4.53e+00	2.63e+00	6.93e+00	7.26e+00	2.74e+00	7.5e+00	2.17e-03
F ₇	Best	4.18e-02	2.75e-04	1.52e-01	3.14e-03	1.13e-03	2.44e-05	1.19e-05	8.01e-05	1.38e-06	3.28e-06
	Mean	2.01e-01	1.49e-02	3.04e-01	1.48e-02	8.21e-03	5.42e-04	1.76e-04	1.75e-03	3.15e-04	1.56e-04
	Worst	6.13e-01	6.71e-02	6.74e-01	3.95e-02	3.31e-02	1.88e-03	6.62e-04	6.30e-03	1.77e-03	5.27e-04
F ₈	Best	-5.06e+03	-1.26e+04	-8.08e+03	-6.55e+03	-1.00e+04	-9.05e+03	-3.45e+03	-3.49e+03	-4.73e+03	-1.26e+04
	Mean	-4.21e+03	-8.75e+03	-7.29e+03	-5.73e+03	-7.52e+03	-5.78e+03	-2.46e+03	-2.74e+03	-3.57e+03	-1.26e+04
	Worst	-3.58e+03	-7.71e+03	-6.73e+03	-5.19e+03	-5.49e+03	-4.26e+03	-1.25e+03	-2.14e+03	-2.86e+03	-1.25e+04
F ₉	Best	2.02e-01	0.e+00	1.18e+02	1.24e-05	1.67e-06	0.e+00	0.e+00	8.52e-10	0.e+00	0.e+00
	Mean	5.46e+01	7.44e+01	2.02e+02	1.81e+01	7.12e-01	0.e+00	0.e+00	4.64e-02	0.e+00	0.e+00
	Worst	1.69e+02	2.58e+02	2.45e+02	1.9e+02	1.71e+01	0.e+00	0.e+00	2.32e+00	0.e+00	0.e+00
F ₁₀	Best	1.58e-01	4.44e-16	5.52e+00	1.98e-04	1.31e-08	4.44e-16	4.44e-16	1.44e-06	4.44e-16	4.44e-16
	Mean	2.57e+00	1.02e-08	1.58e+01	8.40e-03	9.35e-05	4.44e-16	4.44e-16	5.18e-06	7.98e-15	4.44e-16
	Worst	6.23e+00	6.81e-08	2.e+01	1.68e-01	2.28e-03	4.44e-16	4.44e-16	1.44e-05	4.66e-14	4.44e-16
F ₁₁	Best	5.61e-02	0.e+00	3.23e+00	1.47e-07	1.52e-10	0.e+00	0.e+00	4.42e-11	0.e+00	0.e+00
	Mean	1.26e+00	3.43e-02	7.49e+00	6.78e-04	1.14e-04	0.e+00	0.e+00	1.36e-07	0.e+00	0.e+00
	Worst	5.67e+00	5.21e-01	2.08e+01	7.69e-03	3.43e-03	0.e+00	0.e+00	3.48e-06	0.e+00	0.e+00
F ₁₂	Best	8.87e-01	1.68e-02	1.09e+01	2.76e-02	3.21e-02	1.56e-01	6.31e-01	5.15e-02	1.23e+00	5.39e-09
	Mean	3.7e+03	2.86e+00	2.56e+02	2.06e-01	7.68e-02	3.85e-01	1.02e+00	1.36e-01	1.61e+00	1.21e-06
	Worst	5.85e+04	2.31e+01	3.05e+03	9.47e-01	1.90e-01	6.45e-01	1.44e+00	3.84e-01	1.67e+00	8.39e-06
F ₁₃	Best	2.45e+00	3.51e-01	3.46e+01	4.44e-01	3.97e-01	1.66e+00	2.64e+00	9.36e-01	3.19e-01	2.61e-07
	Mean	3.14e+04	4.26e+00	6.05e+04	9.63e-01	1.01e+00	2.87e+00	2.98e+00	1.22e+00	4.88e-01	1.31e-04
	Worst	3.52e+05	4.47e+01	6.11e+05	1.7e+00	2.62e+00	3.e+00	3.e+00	1.52e+00	7.13e-01	1.18e-03

Table 4: Results of benchmark functions (Continued).

F	Measure	Algorithms									
		SCA	WOA	QSA	NRO	GBO	AEO	BMO	GJO	CDO	NIO
F ₁₄	Best	9.98e-01	9.98e-01	9.98e-01	9.98e-01	9.98e-01	9.98e-01	9.99e-01	9.98e-01	2.07e+00	9.98e-01
	Mean	1.20e+00	2.34e+00	9.98e-01	9.98e-01	1.24e+00	2.12e+00	8.97e+00	6.05e+00	1.14e+01	1.1e+00
	Worst	2.98e+00	1.08e+01	9.98e-01	9.98e-01	3.97e+00	1.08e+01	1.27e+01	1.27e+01	1.27e+01	1.99e+00
F ₁₅	Best	3.31e-04	3.08e-04	3.26e-04	3.40e-04	3.11e-04	3.07e-04	4.30e-04	4.04e-04	3.16e-04	3.90e-04
	Mean	6.08e-04	1.82e-03	6.67e-04	4.83e-04	9.01e-04	3.09e-03	6.49e-03	3.38e-03	3.87e-04	6.66e-04
	Worst	7.56e-04	2.04e-02	1.49e-03	6.12e-04	1.49e-03	2.26e-02	2.74e-02	2.09e-02	5.61e-04	1.53e-03
F ₁₆	Best	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00
	Mean	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-9.57e-01	-1.03e+00	-1.00e+00	-1.03e+00
	Worst	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-1.03e+00	-7.06e-01	-1.03e+00	-1.e+00	-1.03e+00
F ₁₇	Best	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01
	Mean	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	3.98e-01	6.56e-01	3.98e-01	4.19e-01	3.98e-01
	Worst	3.99e-01	3.98e-01	3.98e-01	3.98e-01	3.99e-01	3.98e-01	1.46e+00	3.99e-01	5.43e-01	4.e-01
F ₁₈	Best	3.00e+00	3.e+00	3.e+00	3.e+00	3.00e+00	3.e+00	3.28e+00	3.00e+00	4.2e+00	3.00e+00
	Mean	3.00e+00	4.13e+00	3.e+00	3.e+00	3.00e+00	3.e+00	1.50e+01	3.00e+00	3.41e+01	3.00e+00
	Worst	3.00e+00	3.27e+01	3.e+00	3.e+00	3.00e+00	3.e+00	4.14e+01	3.00e+00	8.42e+01	3.00e+00
F ₁₉	Best	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.81e+00	-3.86e+00	-3.86e+00	-3.86e+00
	Mean	-3.86e+00	-3.85e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.51e+00	-3.86e+00	-3.85e+00	-3.86e+00
	Worst	-3.86e+00	-3.09e+00	-3.86e+00	-3.86e+00	-3.86e+00	-3.86e+00	-2.92e+00	-3.85e+00	-3.82e+00	-3.85e+00
F ₂₀	Best	-3.32e+00	-3.32e+00	-3.32e+00	-3.32e+00	-3.32e+00	-3.32e+00	-2.89e+00	-3.32e+00	-3.32e+00	-3.3e+00
	Mean	-3.27e+00	-3.24e+00	-3.25e+00	-3.32e+00	-3.27e+00	-3.26e+00	-2.11e+00	-3.03e+00	-3.20e+00	-3.12e+00
	Worst	-3.18e+00	-3.08e+00	-3.20e+00	-3.32e+00	-3.19e+00	-3.13e+00	-1.26e+00	-1.93e+00	-3.06e+00	-2.96e+00
F ₂₁	Best	-9.95e+00	-1.02e+01	-1.02e+01	-1.02e+01	-1.02e+01	-1.02e+01	-4.79e+00	-1.02e+01	-8.25e+00	-1.02e+01
	Mean	-8.51e+00	-8.63e+00	-7.94e+00	-1.02e+01	-7.91e+00	-9.54e+00	-3.58e+00	-6.02e+00	-4.98e+00	-1.02e+01
	Worst	-4.36e+00	-2.63e+00	-2.63e+00	-1.02e+01	-4.75e+00	-2.63e+00	-2.26e+00	-3.88e+00	-3.79e+00	-1.01e+01
F ₂₂	Best	-1.03e+01	-1.04e+01	-1.04e+01	-1.04e+01	-1.04e+01	-1.04e+01	-5.11e+00	-1.04e+01	-8.29e+00	-1.04e+01
	Mean	-9.06e+00	-7.60e+00	-8.53e+00	-1.04e+01	-8.76e+00	-9.56e+00	-3.52e+00	-8.25e+00	-4.84e+00	-1.04e+01
	Worst	-5.94e+00	-1.84e+00	-2.75e+00	-1.04e+01	-5.08e+00	-2.77e+00	-2.28e+00	-4.86e+00	-3.51e+00	-1.04e+01
F ₂₃	Best	-1.05e+01	-1.05e+01	-1.05e+01	-1.05e+01	-1.05e+01	-1.05e+01	-4.99e+00	-1.05e+01	-8.57e+00	-1.05e+01
	Mean	-9.30e+00	-6.25e+00	-9.28e+00	-1.05e+01	-7.46e+00	-8.72e+00	-3.67e+00	-8.44e+00	-4.98e+00	-1.05e+01
	Worst	-5.11e+00	-1.68e+00	-2.87e+00	-1.05e+01	-2.42e+00	-2.42e+00	-1.85e+00	-9.47e-01	-3.e+00	-1.05e+01

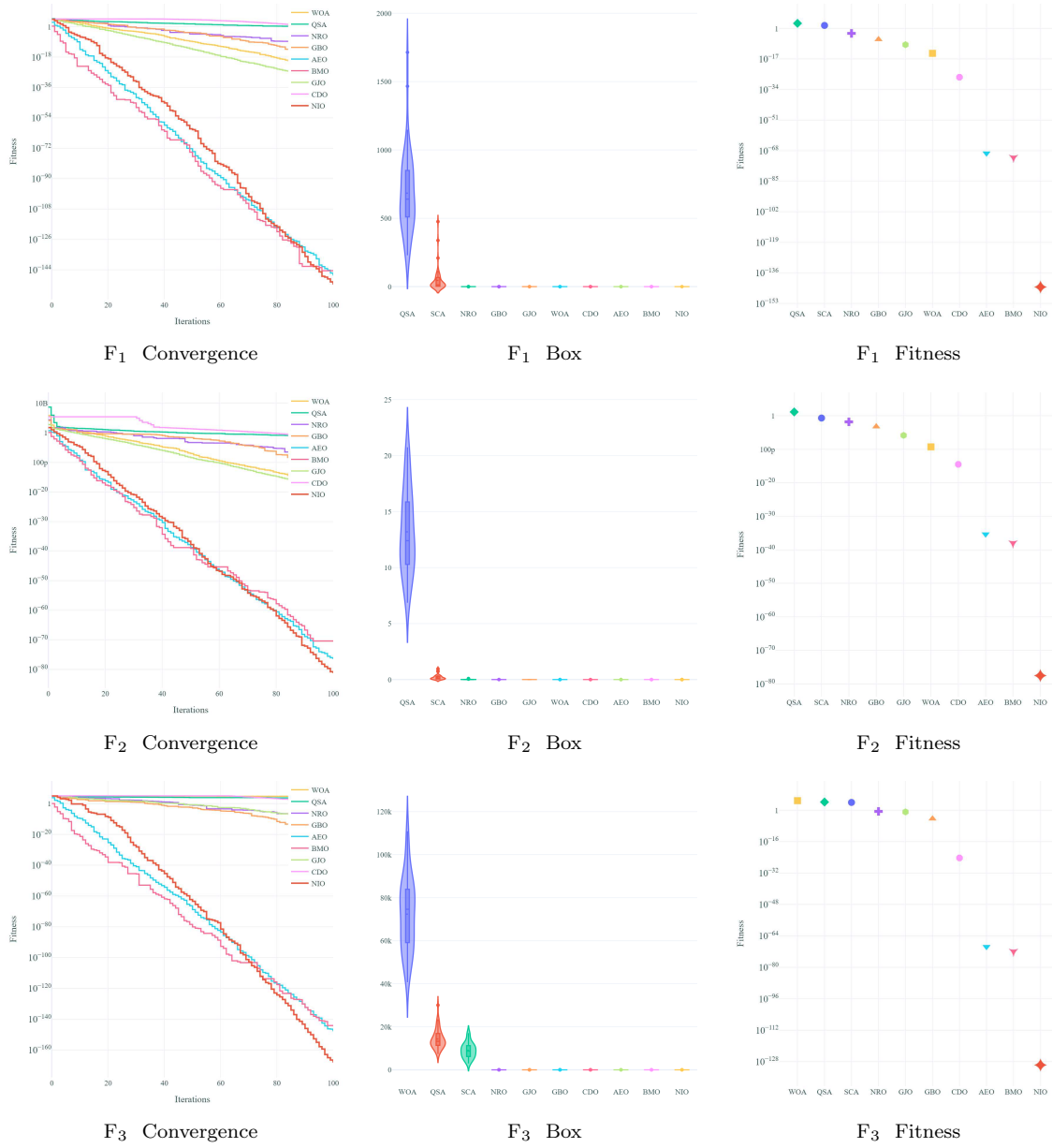
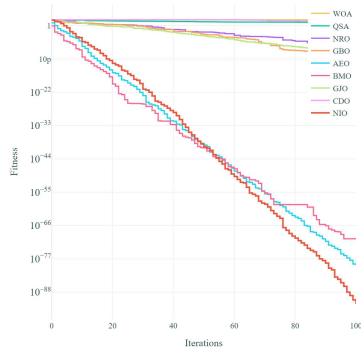


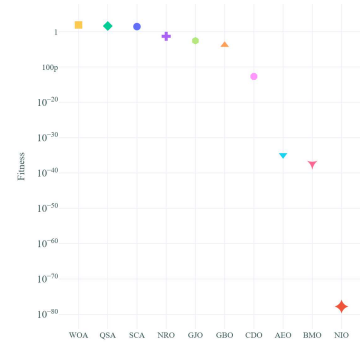
Figure 6: Convergence behavior of the comparative algorithms.



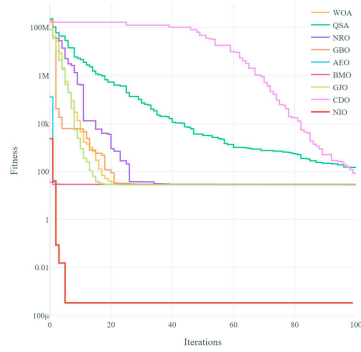
F₄ Convergence



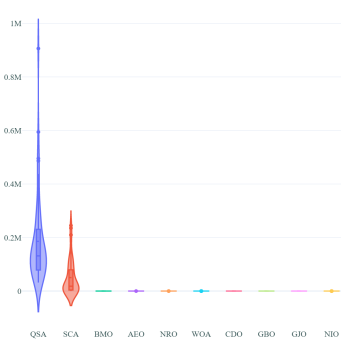
F₄ Box



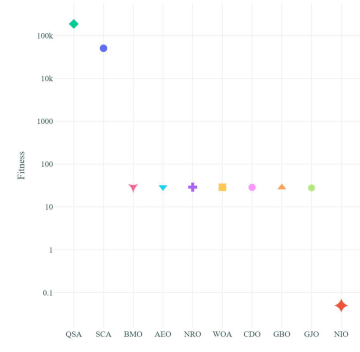
F₄ Fitness



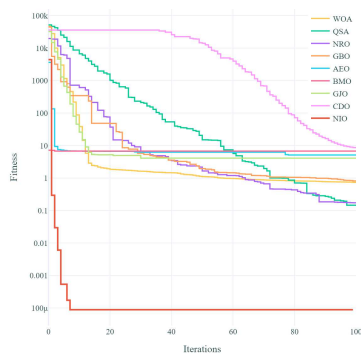
F₅ Convergence



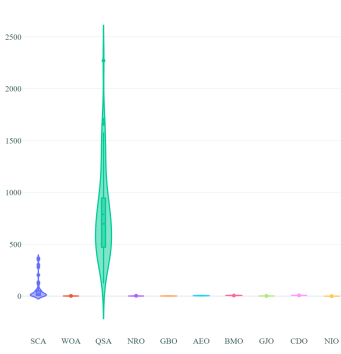
F₅ Box



F₅ Fitness



F₆ Convergence



F₆ Box



F₆ Fitness

Figure 6 (continued)

Table 5: Benchmark functions result where iterations are varied from 50 to 1200.

T	Functions						
	F ₁	F ₅	F ₈	F ₁₂	F ₁₆	F ₂₁	F ₂₃
50	1.466e-65	1.531e-01	-1.256e+04	3.415e-08	-1.031e+00	-1.015e+01	-1.038e+01
100	2.174e-146	1.034e-01	-1.256e+04	4.262e-07	-1.030e+00	-1.015e+01	-1.053e+01
200	9.331e-312	1.434e-02	-1.256e+04	1.333e-06	-1.031e+00	-1.015e+01	-1.054e+01
400	0.0	5.175e-02	-1.256e+04	1.581e-08	-1.031e+00	-1.015e+01	-1.054e+01
600	0.0	4.718e-03	-1.257e+04	2.213e-08	-1.032e+00	-1.015e+01	-1.054e+01
1200	0.0	1.069e-04	-1.257e+04	1.613e-07	-1.032e+00	-1.015e+01	-1.054e+01

Table 6: Benchmark functions result where the population is varied from 25 to 600.

N	Functions						
	F ₁	F ₅	F ₈	F ₁₂	F ₁₆	F ₂₁	F ₂₃
25	1.183e-155	2.363e-03	-1.257e+04	1.618e-05	-1.024e+00	-1.015e+01	-1.054e+01
50	1.118e-168	3.082e-03	-1.256e+04	1.743e-07	-1.032e+00	-1.015e+01	-1.053e+01
100	7.34e-159	9.296e-02	-1.257e+04	3.672e-08	-1.032e+00	-1.015e+01	-1.054e+01
200	3.056e-165	9.255e-04	-1.257e+04	3.632e-07	-1.031e+00	-1.015e+01	-1.054e+01
400	3.944e-163	4.329e-04	-1.257e+04	2.662e-08	-1.032e+00	-1.015e+01	-1.054e+01
600	1.308e-167	8.543e-05	-1.257e+04	2.001e-08	-1.032e+00	-1.015e+01	-1.054e+01

Table 7: The obtained results in the benchmark test functions where different initializers are used.

I	Functions						
	F ₁	F ₅	F ₈	F ₁₂	F ₁₆	F ₂₁	F ₂₃
Random	6.821e-31	2.713e-03	-1.256e+04	4.255e-08	-1.031e+00	-1.014e+01	-1.054e+01
Hypercube	1.226e-31	3.259e-02	-1.257e+04	1.967e-05	-1.031e+00	-1.015e+01	-1.053e+01
Sobol	1.132e-32	1.044e-01	-1.256e+04	2.682e-06	-1.031e+00	-1.014e+01	-1.053e+01
Opposition	2.121e-27	9.372e-01	-1.256e+04	2.323e-07	-1.031e+00	-1.015e+01	-1.053e+01
Halton	1.113e-29	6.782e-01	-1.257e+04	9.790e-07	-1.03e+00	-1.014e+01	-1.054e+01
Sphere	9.956e-29	8.885e-02	-1.253e+04	1.105e-05	-1.031e+00	-1.014e+01	-1.051e+01
Centroid	1.467e-33	1.270e-01	-1.257e+04	3.387e-07	-1.031e+00	-1.015e+01	-1.053e+01
Sine Cosine	1.893e-27	2.626e-03	-1.066e+04	2.704e-07	-1.030e+00	-1.015e+01	-1.053e+01
Hybrid	9.55e-27	6.727e-01	-1.257e+04	5.137e-06	-1.031e+00	-1.006e+01	-1.052e+01
Fibonacci	1.7e-28	3.145e-02	-1.257e+04	1.061e-07	-1.031e+00	-1.015e+01	-1.054e+01

Adaptive parameters, agility (α) and precision (ϵ), dynamically refine the search process from broad exploration to focused exploitation. These parameters significantly enhance NIO's flexibility across diverse optimization landscapes, emphasizing the importance of parameter tuning for problem-specific performance.

4.6 Run-Time Complexity

The computational complexity analysis in Table 8 reveals that while no single algorithm consistently dominates across all benchmark functions, the **BMO** algorithm generally exhibits the lowest computation times, particularly for simpler problems (F_1 – F_{13}). In contrast, the **NIO** algorithm, though not always the fastest, consistently demonstrates competitive performance—often ranking among the top algorithms in terms of execution time. It strikes a strong balance between speed and robustness, especially on more complex functions such as F_{14} and F_{21} – F_{23} , where it outperforms other methods. Notably, **NIO** achieved a significantly lower computation time of 1.909 seconds for F_{14} compared to **AEO**, which recorded 10.375 seconds. For the most computationally demanding functions (F_{21} – F_{23}), **NIO** maintained exceptional efficiency—for instance, achieving 1.841 seconds on F_{22} . These results highlight **NIO**’s balanced performance, combining computational efficiency with robust optimization capabilities, particularly in complex problem settings.

Table 8: Benchmark functions time complexity.

F_x	Algorithms time complexity (s)									
	SCA	WOA	QSA	NRO	GBO	AEO	BMO	GJO	CDO	NIO
F_1	4.559	0.932	5.812	7.569	3.050	1.647	0.899	2.852	1.183	1.913
F_2	4.393	1.059	5.795	7.509	2.912	1.650	0.809	2.779	1.239	1.948
F_3	6.016	2.458	10.136	11.924	4.310	4.602	2.296	4.280	2.614	1.918
F_4	4.355	0.921	5.649	7.270	2.835	1.571	0.687	2.874	1.190	1.848
F_5	4.691	1.089	6.162	7.861	3.186	1.856	0.793	2.863	1.376	1.910
F_6	4.557	0.944	5.767	7.113	3.008	1.567	0.630	2.653	1.147	1.941
F_7	4.572	1.077	5.850	7.772	3.084	1.855	0.850	2.808	1.382	1.878
F_8	4.435	0.964	5.383	6.867	2.668	1.465	0.659	2.632	1.106	1.902
F_9	4.385	0.986	5.516	7.464	2.832	1.639	0.687	2.638	1.214	2.016
F_{10}	4.240	1.096	5.829	7.485	2.911	1.835	0.823	2.751	1.268	1.858
F_{11}	4.350	1.050	5.855	7.484	2.917	1.871	0.779	2.763	1.350	1.925
F_{12}	4.602	1.436	6.597	8.437	3.268	2.400	1.167	2.964	1.605	1.870
F_{13}	4.681	1.318	6.952	8.567	3.302	2.467	1.215	3.050	1.533	1.970
F_{14}	5.751	5.447	15.486	18.363	7.442	10.375	5.103	5.152	5.653	1.909
F_{15}	1.716	1.101	3.072	5.361	2.884	1.735	0.750	1.027	1.250	1.954
F_{16}	1.150	0.796	2.062	4.625	2.665	1.435	0.616	0.729	1.047	1.935
F_{17}	1.290	0.868	2.282	4.509	2.712	1.345	0.689	0.805	0.946	1.981
F_{18}	1.197	0.871	2.295	4.695	2.620	1.403	0.542	0.789	0.967	2.047
F_{19}	1.829	1.323	3.706	6.208	3.181	2.265	1.067	1.301	1.427	1.941
F_{20}	2.190	1.344	3.838	6.480	3.211	2.329	1.156	1.433	1.483	1.865
F_{21}	3.956	3.028	9.315	12.118	5.252	5.967	2.786	3.164	3.318	1.927
F_{22}	4.536	3.895	11.258	14.476	6.117	7.501	3.517	3.830	4.131	1.841
F_{23}	5.914	4.890	14.932	18.084	7.385	9.711	4.745	5.357	5.143	1.900

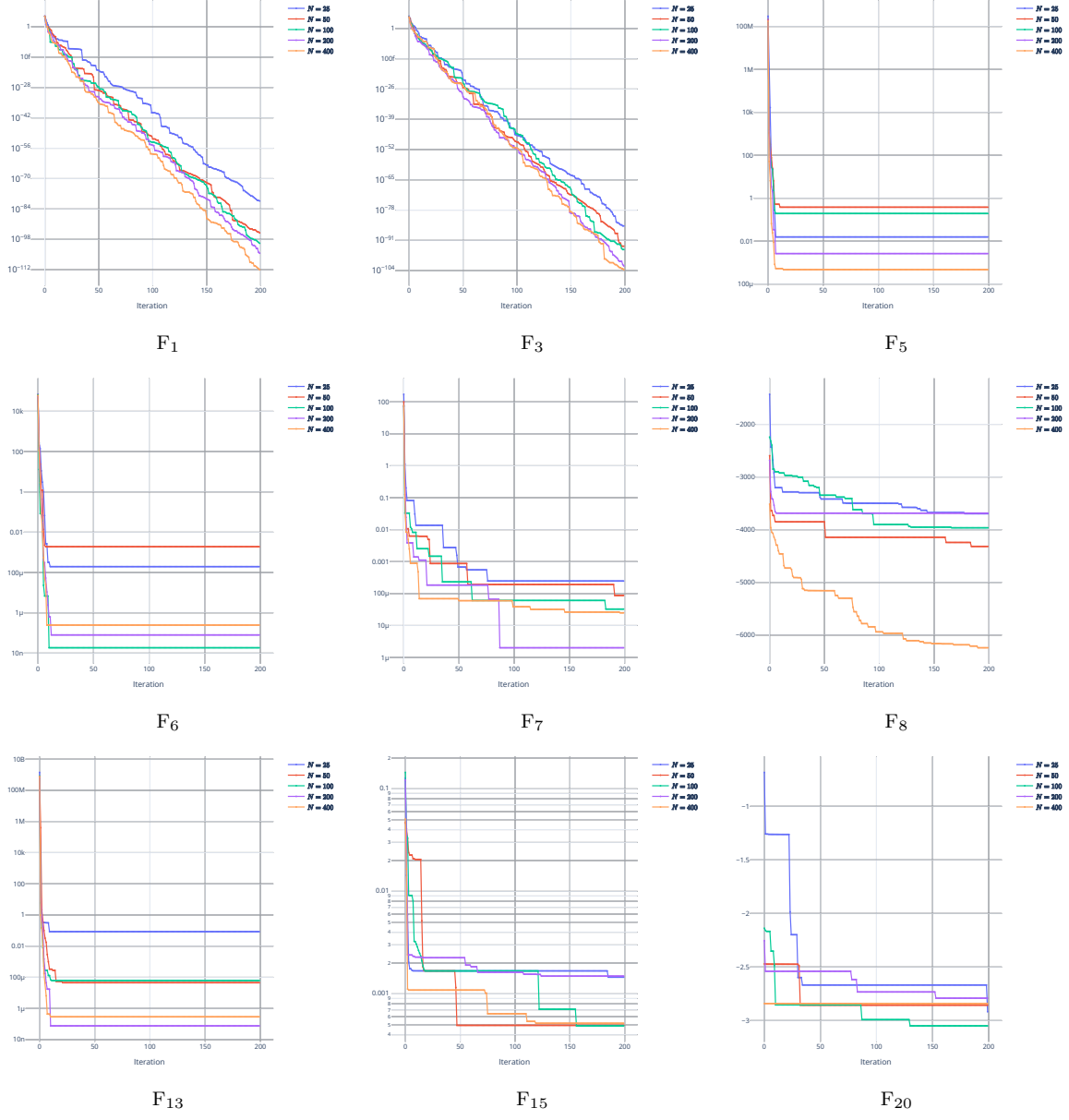


Figure 7: The influence of the NIO population size (N) tested on various classical test functions.

4.7 Performance Analysis on CEC Composite Benchmark Suites

To evaluate the proposed algorithm’s efficacy under more rigorous conditions, the experimental study was extended to include the CEC 2019 and CEC 2022 benchmark suites. These recently developed test suites introduce greater complexity through shifted, rotated, and hybrid composition functions, which rigorously challenge an optimizer’s ability to escape local optima. Testing on these modern benchmarks ensures that the exploration-exploitation balance of the proposed method remains effective even in high-dimensional and non-convex environments. The results, detailed below, highlight the algorithm’s adaptability compared to other established metaheuristics.

The performance evaluation across the CEC 2019 test suite reveals distinct hierarchical groupings among the ten algorithms. NIO emerges as the most robust general-purpose optimizer, achieving the lowest mean values on six of the ten functions (F_5 through F_{10}) and consistently ranking among the top performers elsewhere. Its superiority is particularly pronounced on the hybrid composition functions (F_6 , F_9) where it maintains orders-of-magnitude advantages over competitors. NRO demonstrates exceptional specialization on the unimodal and simple multimodal functions, delivering the optimal mean performance on F_7 (1.0014) and competitive results on F_8 and F_9 . However, its performance degrades on more complex compositions, suggesting limited scalability. The AEO, BMO, and CDO algorithms exhibit an interesting dichotomy: they achieve perfect theoretical optima on the simplest F_1 function (exact 1.0 values), yet catastrophically fail on moderate-complexity functions, with BMO and CDO producing mean values exceeding 10^4 on F_4 and F_9 .

The SCA and WOA metaheuristics display consistent mid-tier performance, never achieving best-in-class results but avoiding the extreme failures of weaker algorithms. Their exploration-exploitation balance appears adequate for low-dimensional problems but insufficient for precise convergence, as evidenced by mean values 2-3 orders of magnitude worse than leaders on unimodal functions. QSA shows promise on specific functions (lowest mean on F_5) but lacks consistency, while GBO and GJO occupy a middle ground with occasional competitive results but no dominant performance patterns.

The CEC 2022 results reveal a more dramatic performance stratification, with NRO establishing clear dominance across nine of twelve functions, particularly excelling in the high-dimensional expanded problems (F_2 , F_9 – F_{12}) where it achieves near-theoretical minima. Its mean error of 2575 on F_6 represents a 99.96% improvement over the second-best algorithm, showcasing remarkable convergence precision. NIO consistently functions as the secondary workhorse, delivering the second-lowest means on most functions and capturing the best performance on F_7 . This establishes NIO-NRO as a complementary pair for robust optimization suites. In stark contrast, BMO exhibits catastrophic failures on composite functions, producing astronomical mean values of 7.6×10^8 on F_6 and 2.0×10^{13} on F_8 —results that invalidate its use for complex optimization.

Table 9: Results of CEC 2019 benchmark functions.

F	Measure	Algorithms									
		SCA	WOA	QSA	NRO	GBO	AEO	BMO	GJO	CDO	NIO
F ₁	Best	4.26e+04	1.61e+06	6.47e+04	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	2.46e+03
	Mean	8.26e+05	3.29e+07	9.45e+05	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.01e+06
	Worst	2.50e+06	6.82e+07	6.20e+06	1.01e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	3.81e+06
F ₂	Best	1.08e+02	1.30e+03	1.13e+02	9.31e+00	4.37e+00	4.34e+00	4.98e+00	5.14e+00	5.00e+00	4.46e+01
	Mean	3.00e+02	4.05e+03	1.90e+02	3.79e+01	4.65e+00	4.60e+00	5.00e+00	3.13e+01	5.00e+00	3.02e+02
	Worst	4.50e+02	7.81e+03	3.20e+02	7.43e+01	5.28e+00	5.00e+00	5.00e+00	1.85e+02	5.00e+00	6.71e+02
F ₃	Best	3.71e+00	2.71e+00	4.87e+00	5.32e+00	1.65e+00	3.43e+00	6.87e+00	1.05e+00	5.62e+00	4.60e+00
	Mean	5.39e+00	5.89e+00	7.12e+00	6.72e+00	4.54e+00	4.96e+00	9.08e+00	4.97e+00	6.59e+00	8.05e+00
	Worst	7.16e+00	8.60e+00	9.30e+00	7.88e+00	1.07e+01	7.31e+00	1.07e+01	7.83e+00	7.60e+00	1.17e+01
F ₄	Best	1.13e+02	5.98e+02	7.49e+00	1.82e+01	6.12e+01	2.81e+02	1.34e+04	5.92e+01	1.38e+04	1.10e+01
	Mean	1.56e+02	1.96e+03	2.80e+01	2.44e+01	9.70e+01	5.94e+02	2.02e+04	4.18e+02	2.08e+04	1.65e+01
	Worst	1.90e+02	4.04e+03	3.60e+01	2.97e+01	1.53e+02	1.35e+03	2.75e+04	1.02e+03	2.43e+04	2.13e+01
F ₅	Best	1.41e+00	1.47e+00	1.05e+00	1.09e+00	1.18e+00	1.57e+00	4.00e+00	1.17e+00	3.93e+00	1.05e+00
	Mean	1.71e+00	2.02e+00	1.17e+00	1.12e+00	1.41e+00	2.08e+00	5.45e+00	1.50e+00	3.98e+00	1.14e+00
	Worst	1.85e+00	3.30e+00	1.34e+00	1.15e+00	1.60e+00	3.33e+00	6.57e+00	1.83e+00	4.03e+00	1.52e+00
F ₆	Best	8.54e+00	7.43e+00	9.07e+00	7.77e+00	1.14e+01	6.92e+00	1.28e+01	5.97e+00	1.04e+01	2.90e+00
	Mean	1.12e+01	9.76e+00	1.03e+01	1.00e+01	1.17e+01	9.64e+00	1.35e+01	8.45e+00	1.17e+01	4.66e+00
	Worst	1.20e+01	1.14e+01	1.13e+01	1.12e+01	1.22e+01	1.07e+01	1.47e+01	9.38e+00	1.28e+01	7.35e+00
F ₇	Best	5.34e+00	1.43e+00	1.00e+00	1.00e+00	1.33e+00	4.24e+00	8.96e+02	5.60e+00	8.96e+02	1.00e+00
	Mean	8.47e+00	1.06e+02	1.16e+02	1.00e+00	5.72e+01	1.43e+02	1.33e+03	1.12e+02	1.32e+03	7.67e+01
	Worst	1.47e+01	4.35e+02	5.97e+02	1.00e+00	2.98e+02	4.77e+02	1.80e+03	3.30e+02	1.62e+03	2.95e+02
F ₈	Best	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.55e+00	1.01e+00	1.21e+00	1.00e+00
	Mean	1.00e+00	1.01e+00	1.00e+00	1.00e+00	1.00e+00	1.02e+00	1.72e+00	1.05e+00	1.22e+00	1.00e+00
	Worst	1.01e+00	1.06e+00	1.00e+00	1.00e+00	1.00e+00	1.04e+00	1.88e+00	1.06e+00	1.24e+00	1.00e+00
F ₉	Best	4.30e+00	1.68e+00	1.25e+00	1.27e+00	1.65e+00	4.60e+00	5.05e+02	7.43e+00	7.22e+02	1.06e+00
	Mean	6.45e+00	1.53e+01	1.32e+00	1.34e+00	2.09e+00	1.36e+01	6.88e+02	2.21e+01	1.05e+03	1.18e+00
	Worst	8.49e+00	5.72e+01	1.37e+00	1.41e+00	2.54e+00	2.44e+01	9.07e+02	5.88e+01	1.31e+03	1.28e+00
F ₁₀	Best	1.28e+01	2.10e+01	2.11e+01	1.54e+01	2.12e+01	1.86e+01	2.16e+01	1.42e+01	2.15e+01	2.10e+01
	Mean	2.02e+01	2.12e+01	2.13e+01	2.07e+01	2.14e+01	2.10e+01	2.18e+01	1.99e+01	2.16e+01	2.10e+01
	Worst	2.17e+01	2.14e+01	2.15e+01	2.14e+01	2.16e+01	2.17e+01	2.20e+01	2.14e+01	2.17e+01	2.12e+01

The differential evolution variants QSA and NIO demonstrate exceptional reliability on simpler unimodal and basic multimodal functions (F₁–F₃), achieving values within 0.001% of optima, but their performance gaps widen as problem complexity increases. SCA and WOA maintain their mid-tier status, though SCA’s best-in-class minimum on F₄ (816.81) suggests occasional exploitation advantages. The CDO algorithm’s performance mirrors BMO’s trajectory, with mean values exceeding 10⁸ on composite functions, indicating fundamental difficulties with dimensionality and landscape complexity. Overall, only NRO, NIO, and QSA demonstrate reliability across the entire benchmark spectrum, while traditional metaheuristics (SCA, WOA, GBO, GJO) show diminishing returns on the most challenging functions.

Table 10: Results of CEC 2022 benchmark functions.

F	Measure	Algorithms									
		SCA	WOA	QSA	NRO	GBO	AEO	BMO	GJO	CDO	NIO
F ₁	Best	3.38e+02	9.52e+02	3.00e+02	3.00e+02	3.02e+02	3.87e+02	7.11e+03	4.70e+02	9.97e+03	3.00e+02
	Mean	4.16e+02	3.43e+03	3.00e+02	3.00e+02	3.10e+02	6.13e+02	9.72e+03	8.31e+02	1.31e+04	3.00e+02
	Worst	5.01e+02	9.87e+03	3.01e+02	3.00e+02	3.42e+02	9.16e+02	1.25e+04	3.90e+03	1.39e+04	3.00e+02
F ₂	Best	4.10e+02	4.13e+02	4.05e+02	4.00e+02	4.02e+02	4.04e+02	1.19e+03	4.50e+02	8.40e+02	4.00e+02
	Mean	4.31e+02	4.50e+02	4.15e+02	4.01e+02	4.21e+02	4.69e+02	2.81e+03	4.72e+02	8.91e+02	4.14e+02
	Worst	4.49e+02	4.83e+02	4.76e+02	4.02e+02	4.72e+02	5.14e+02	4.80e+03	5.32e+02	9.44e+02	4.76e+02
F ₃	Best	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.01e+02	6.00e+02	6.00e+02	6.00e+02
	Mean	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.01e+02	6.00e+02	6.00e+02	6.00e+02
	Worst	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.01e+02	6.00e+02	6.00e+02	6.00e+02
F ₄	Best	8.17e+02	8.49e+02	8.26e+02	8.17e+02	8.17e+02	8.26e+02	9.04e+02	8.18e+02	8.46e+02	8.26e+02
	Mean	8.33e+02	9.06e+02	8.39e+02	8.24e+02	8.42e+02	8.47e+02	9.31e+02	8.44e+02	8.58e+02	8.51e+02
	Worst	8.57e+02	9.96e+02	8.52e+02	8.35e+02	8.93e+02	8.70e+02	9.63e+02	8.66e+02	8.80e+02	8.78e+02
F ₅	Best	9.00e+02	9.01e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.02e+02	9.00e+02	9.02e+02	9.00e+02
	Mean	9.00e+02	9.03e+02	9.00e+02	9.00e+02	9.00e+02	9.01e+02	9.04e+02	9.00e+02	9.03e+02	9.01e+02
	Worst	9.00e+02	9.05e+02	9.01e+02	9.00e+02	9.01e+02	9.02e+02	9.05e+02	9.01e+02	9.06e+02	9.03e+02
F ₆	Best	4.58e+04	2.84e+03	2.54e+03	1.99e+03	1.10e+04	9.90e+03	1.04e+08	1.41e+04	1.04e+08	2.04e+04
	Mean	9.29e+04	3.78e+04	2.03e+04	2.58e+03	3.91e+04	3.11e+04	7.62e+08	3.20e+04	7.51e+08	4.43e+04
	Worst	1.52e+05	9.63e+04	5.46e+04	3.27e+03	6.39e+04	6.51e+04	2.40e+09	7.32e+04	1.55e+09	6.91e+04
F ₇	Best	2.05e+03	2.03e+03	2.03e+03	2.03e+03	2.04e+03	2.03e+03	2.49e+03	2.04e+03	2.65e+03	2.01e+03
	Mean	2.06e+03	2.23e+03	2.03e+03	2.04e+03	2.05e+03	2.09e+03	3.07e+03	2.20e+03	4.13e+03	2.02e+03
	Worst	2.07e+03	2.48e+03	2.04e+03	2.04e+03	2.07e+03	2.37e+03	3.96e+03	2.32e+03	5.53e+03	2.03e+03
F ₈	Best	2.26e+03	2.37e+03	2.22e+03	2.22e+03	2.22e+03	2.25e+03	1.20e+08	3.90e+03	2.34e+03	2.21e+03
	Mean	2.32e+03	4.35e+03	2.23e+03	2.23e+03	2.39e+03	2.52e+03	2.05e+13	4.17e+03	2.83e+03	2.22e+03
	Worst	2.46e+03	7.39e+03	2.24e+03	2.28e+03	3.48e+03	3.12e+03	2.04e+14	4.58e+03	3.34e+03	2.22e+03
F ₉	Best	2.31e+03	2.33e+03	2.30e+03	2.30e+03	2.31e+03	2.36e+03	3.07e+03	2.71e+03	3.07e+03	2.30e+03
	Mean	2.32e+03	2.70e+03	2.64e+03	2.30e+03	2.60e+03	2.64e+03	3.78e+03	2.75e+03	3.32e+03	2.56e+03
	Worst	2.32e+03	2.90e+03	2.94e+03	2.30e+03	2.67e+03	2.78e+03	4.35e+03	2.80e+03	3.48e+03	2.67e+03
F ₁₀	Best	2.62e+03	2.63e+03	2.62e+03	2.62e+03	2.62e+03	2.63e+03	2.83e+03	2.62e+03	2.79e+03	2.62e+03
	Mean	2.63e+03	2.82e+03	2.62e+03	2.62e+03	2.65e+03	2.73e+03	3.54e+03	2.70e+03	2.82e+03	2.69e+03
	Worst	2.63e+03	3.29e+03	2.63e+03	2.62e+03	2.76e+03	2.81e+03	4.40e+03	2.78e+03	3.00e+03	2.75e+03
F ₁₁	Best	2.60e+03	2.60e+03	2.60e+03	2.60e+03	2.60e+03	2.60e+03	2.99e+03	2.61e+03	2.85e+03	2.60e+03
	Mean	2.60e+03	2.71e+03	2.60e+03	2.60e+03	2.78e+03	2.92e+03	3.89e+03	2.71e+03	2.87e+03	2.69e+03
	Worst	2.61e+03	3.52e+03	2.63e+03	2.60e+03	3.48e+03	3.87e+03	4.85e+03	3.52e+03	2.88e+03	3.48e+03
F ₁₂	Best	2.87e+03	2.88e+03	2.87e+03	2.87e+03	2.86e+03	2.87e+03	3.18e+03	2.87e+03	3.18e+03	2.87e+03
	Mean	2.88e+03	2.97e+03	2.87e+03	2.87e+03	2.87e+03	2.91e+03	3.33e+03	2.88e+03	3.33e+03	2.87e+03
	Worst	2.89e+03	3.10e+03	2.88e+03	2.87e+03	2.87e+03	2.96e+03	3.50e+03	2.91e+03	3.58e+03	2.87e+03

4.8 Applications in Engineering Problems

Metaheuristics effectively emulate natural phenomena to navigate complex solution spaces in engineering optimization problems where traditional methods fall short. We demonstrate NIO's applicability across several critical engineering domains, emphasizing mechanical and structural optimization tasks.

In mechanical engineering, the tension/compression spring design problem (Section 4.8.1) optimizes spring dimensions, balancing mechanical performance, weight, and cost. The pressure vessel design (Section 4.8.2) ensures safe high-pressure operation with optimal material usage. Additionally, the speed reducer problem (Section 4.8.3) identifies efficient gear configurations for reliable performance.

Structural optimization applications include the three-bar truss problem (Section 4.8.4), minimizing weight while maintaining structural integrity. The welded beam design (Section 4.8.5) optimizes geometry and weld size, balancing strength, stability, and cost. Cantilever beam design (Section 4.8.6) determines dimensions that minimize deflection and stress under load.

Mechanical systems further benefit from optimization, such as the multiple-disc clutch brake problem (Section 4.8.7), optimizing system performance, durability, and cost-efficiency, and the piston lever design (Section 4.8.8), achieving desired motion characteristics and durability through optimal lever dimensions and materials.

4.8.1 Tension/Compression Spring Design Problem

The tension/compression spring design optimization seeks to minimize weight and material costs while satisfying operational constraints on shear stress, deflection, and resonance frequencies (Fig. 8). Critical parameters include wire diameter (d), mean coil diameter (D), and the number of active coils (N), which determine spring performance and durability [19].

Comparative results (Table 11) demonstrate NIO's superior performance, achieving the lowest cost (0.01267) among tested metaheuristics. Optimal parameter values identified by NIO ($d = 0.052$, $D = 0.361$, $N = 11.041$) highlight its efficacy in navigating complex search landscapes. Competing algorithms, such as QSA (0.01273) and NRO (0.01271), provided near-optimal results but did not match NIO's capability. This superior performance underscores NIO's balanced exploration-exploitation strategy, making it particularly effective for complex engineering optimization tasks.

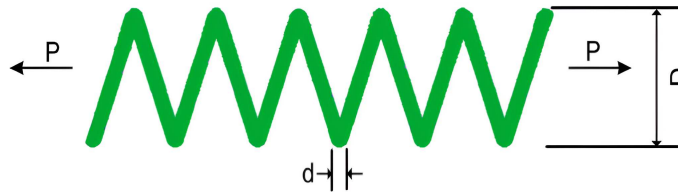


Figure 8: Tension/compression spring design problem.

Table 11: Comparative algorithms results for solving the tension/compression spring design problem.

Algorithm	Optimal Values			Cost
	d	D	N	
SCA	0.053	0.396	9.431	0.01287
WOA	0.059	0.547	5.179	0.01347
QSA	0.051	0.349	11.793	0.01273
NRO	0.052	0.362	10.996	0.01271
GBO	0.05	0.314	14.526	0.0129
AEO	0.058	0.521	5.649	0.01329
BMO	0.075	0.574	8.086	33759289
GJO	0.063	0.682	3.535	0.01492
CDO	0.050	0.312	15.000	0.01324
NIO	0.052	0.361	11.041	0.01267

4.8.2 Pressure Vessel Design Problem

The pressure vessel design problem involves optimizing vessel dimensions and materials to safely contain high-pressure fluids, crucial in industries such as petrochemicals and aerospace (Fig. 9). Key parameters include shell thickness (T_s), head thickness (T_h), inner radius (R), and cylindrical length (L), subject to stringent constraints (g_1 – g_4) ensuring structural integrity and practical feasibility [7, 26, 62].

Comparative results (Table 12) confirm NIO’s strong performance, achieving the lowest cost of 6077.3404 with optimal dimensions: $T_s = 13.0$, $T_h = 7.0$, $R = 42.018$, and $L = 177.971$. Though competitive algorithms like AEO (6093) provided close outcomes, NIO’s balanced exploration-exploitation consistently outperformed alternatives such as WOA (22929) and BMO (13192). These results highlight NIO’s robust capability to identify optimal material thickness and explore diverse solution spaces effectively.

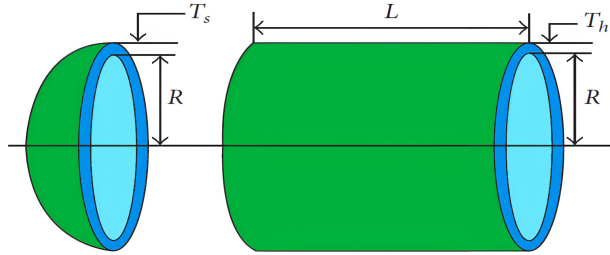


Figure 9: Pressure vessel design problem.

4.8.3 Speed Reducer Problem

The speed reducer optimization aims to efficiently design gear systems that reliably adjust motor speeds and torques for machinery (Fig. 10). It involves seven critical parameters: face width (x_1),

Table 12: Comparative results for the pressure vessel design problem.

Algorithm	Optimal Values				Cost
	T_s	T_h	R	L	
SCA	13.0	7.0	42.032	180.172	6131
WOA	16.0	67.0	49.911	99.055	22929
NRO	14.0	7.0	45.295	141.197	6109
GB0	13.0	7.0	42.066	181.890	6177
QSA	14.0	7.0	45.302	140.926	6103
AEO	14.0	7.0	45.337	140.359	6093
BMO	21.0	18.0	56.969	91.424	13192
GJO	13.0	7.0	41.637	183.110	6132
CDO	15.0	7.0	42.575	200.0	7677
NIO	13.0	7.0	42.018	177.971	6077

tooth module (x_2), pinion tooth number (x_3), shaft lengths (x_4, x_5), and shaft diameters (x_6, x_7) [7, 26].

The results of this problem, as presented in Table 13, demonstrate that the NIO achieves a lower total cost (2769) than all other metaheuristic algorithms except the GBO (2777), outperforming the remaining nine algorithms.

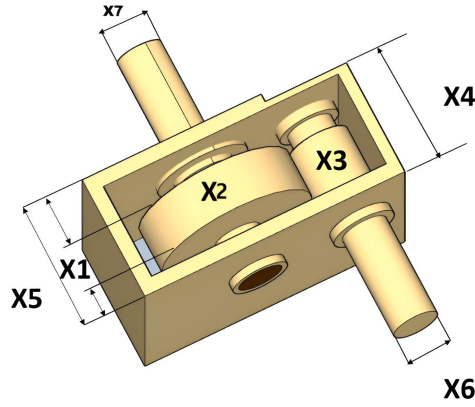


Figure 10: Speed reducer design problem.

4.8.4 Three Bar Truss Problem

The three-bar truss problem is a classic structural optimization challenge where the goal is to determine the optimal configuration of bars in a truss structure to minimize weight while maintaining structural integrity. Trusses are widely used in bridges, buildings, and other structures for their efficiency in bearing loads [7, 26]. The three-bar truss design specifically focuses on optimizing two parameters, the length (x_1), and cross-sectional area (x_2) to achieve this weight minimization (Fig. 11).

The objective is to find the optimal design that minimizes the total weight of the truss while ensuring it can support the applied loads without exceeding stress and deflection limits. Constraints typically include limits on axial stress, displacement, and stability against buckling. Comparative results (Table 14) confirm NIO's effectiveness relative to other algorithms.

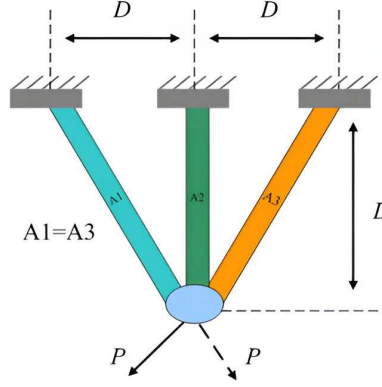


Figure 11: Three-bar truss design problem.

4.8.5 Welded Beam Design Problem

The welded beam design optimization seeks to minimize material usage and cost by optimizing beam geometry and weld dimensions while ensuring structural strength and stability [5]. The critical variables include weld thickness (h), beam height (t), beam thickness (b), and beam length (l), constrained by limits on displacement, stress, and buckling (Fig. 12).

Comparative analysis (Table 15) reveals that NIO achieves the lowest optimal cost (1.614) among evaluated metaheuristics. Optimal parameters found were $h = 0.213$, $l = 3.508$, $t = 7.730$, and $b = 0.221$, demonstrating NIO's efficacy in managing trade-offs between structural demands and cost-efficiency.

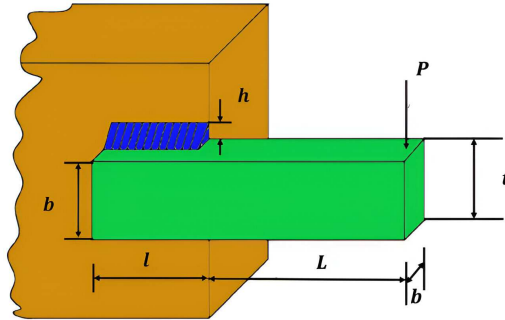


Figure 12: Welded beam design problem.

4.8.6 Cantilever Beam Design Problem

The cantilever beam design problem involves optimizing the beam dimensions that is fixed at one end and free at the other to minimize deflection and stress under applied loads. Its main intention is to locate five optimal structural variables $[x_1 : x_5]$, (Fig. 13), to minimize the total weight of a cantilever beam when meeting the load capacity requirements. The aim is to find optimal dimensions that minimize deflection and stress while ensuring that the beam can support the applied loads without failing.

The results in Table 16 indicate the competitive performance of NIO, achieving an optimal cost (1.343) equivalent to the best performing QSA algorithm. The obtained variables ($x_1 = 6.181$, $x_2 = 5.043$, $x_3 = 4.645$, $x_4 = 3.446$, $x_5 = 2.221$) suggest balanced optimization. Algorithms like GJO (1.341) and QSA (1.343) closely rival NIO, whereas others, such as WOA (6.203), show significant performance gaps due to convergence issues.

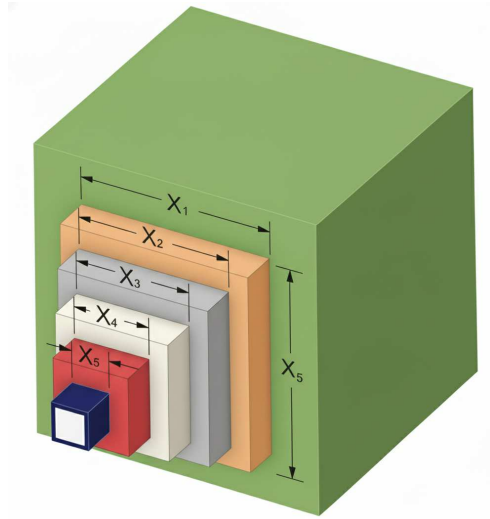


Figure 13: Cantilever beam design problem [14].

4.8.7 Multiple-disc Clutch Brake Problem

The multiple-disc clutch brake optimization addresses a complex mechanical system that is shown in Fig. 14 by minimizing design costs related to disc thickness (x_3), inner and outer radii (x_1, x_2), actuating force (x_4), and friction surface number (x_5). This problem demands an intricate balancing of performance and cost considerations.

Results (Table 17) show NIO outperforming competitors with the lowest cost (0.23521). Optimal parameters identified were $x_1 = 70.0$, $x_2 = 90.0$, $x_3 = 1.0$, $x_4 = 239.093$, and $x_5 = 2.0$. Other algorithms, such as QSA and AEO (0.23524), provided comparable but less optimal solutions, whereas WOA and BMO exhibited significantly inferior performances (0.33081).

Table 13: Comparative results for the speed reducer design problem.

Algorithm	Optimal Values							Cost
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
SCA	3.542	0.708	17.115	7.868	7.966	3.656	5.325	3131
WOA	3.500	0.7	17.0	7.744	7.885	3.460	5.0	2804
QSA	3.500	0.7	17.0	7.300	7.800	3.350	5.0	2769
NRO	3.515	0.7	17.009	7.422	7.849	3.363	5.002	2784
GBO	3.517	0.7	17.0	7.33	7.8	3.353	5.0	2777
AEO	3.502	0.7	17.0	7.300	8.086	3.350	5.0	2775
BMO	3.553	0.713	21.503	8.047	8.285	3.446	5.490	315281
GJO	3.504	0.7	17.0	7.840	7.800	3.402	5.0	2791
CDO	3.600	0.7	17.0	7.300	7.800	3.369	5.0	2812
NIO	3.500	0.7	17.0	7.3	7.800	3.350	5.0	2769

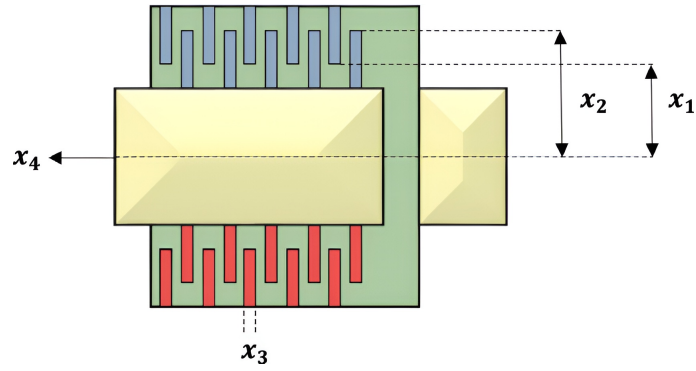


Figure 14: multiple-disc clutch brake problem [14].

Table 14: Benchmarking results for solving the three-bar truss design problem.

Algorithm	Optimal Values		Cost
	x_1	x_2	
SCA	0.784	0.422	263.913
WOA	0.738	0.576	266.234
QSA	0.789	0.407	263.896
NRO	0.789	0.407	263.896
GB0	0.793	0.393	263.937
AEO	0.788	0.411	263.896
BMO	0.723	0.678	272.172
GJO	0.816	0.360	266.939
CDO	0.789	0.407	263.926
NIO	0.790	0.405	263.817

Table 15: Comparative results of the welded beam design problem.

Algorithm	Optimal values				Cost
	h	l	t	b	
SCA	0.258	3.060	8.071	0.259	1.943
WOA	0.406	3.473	4.855	0.713	3.543
NRO	0.167	4.032	9.045	0.206	1.740
QSA	0.208	3.127	8.999	0.209	1.698
GB0	0.127	5.493	9.095	0.203	1.833
AEO	0.125	7.766	7.799	0.276	2.390
BMO	0.557	3.058	6.348	0.702	4.707
GJO	0.169	3.961	9.206	0.206	1.763
CDO	0.181	3.531	9.762	0.211	1.865
NIO	0.213	3.508	7.730	0.221	1.614

4.8.8 Piston Lever Design Problem

The piston lever design problem involves optimizing the dimensions and materials of a lever mechanism to achieve desired motion characteristics and durability. Piston levers are used in various mechanical systems to convert linear motion into rotational motion or vice versa. Parameters include the length and cross-sectional area of the lever, and the position of the pivot point.

The objective is to find the optimal design that maximizes efficiency and durability while minimizing weight and cost. The main objective of this problem is to locate the piston components, $H(x_1)$, $B(x_2)$, $D(x_3)$, and $X(x_4)$, by minimizing the oil volume when the lever of the piston is lifted up from 0° to 45° , as shown in Fig. 15.

The results presented in Table 18 provide a comparative analysis of various metaheuristic algorithms applied to the piston lever design problem. Among the algorithms compared, the NIO algorithm demonstrates the best performance, achieving the lowest cost of 1.05. This result high-

Table 16: Comparative results for solving the cantilever beam design problem.

Algorithm	Optimal Values					Cost
	x_1	x_2	x_3	x_4	x_5	
SCA	5.918	5.433	4.695	3.752	1.860	1.351
WOA	16.298	33.598	5.710	1.994	41.802	6.203
QSA	6.067	5.320	4.348	3.714	2.079	1.343
NRO	6.244	5.035	4.659	3.503	2.155	1.348
GBO	6.182	4.954	4.109	4.091	2.787	1.380
AEO	5.427	6.723	5.191	2.887	2.429	1.414
BMO	6.260	6.118	4.452	2.825	6.903	1.657
GJO	6.084	5.360	4.411	3.503	2.124	1.341
CDO	5.750	5.327	4.228	4.654	2.278	1.388
NIO	6.181	5.043	4.645	3.446	2.221	1.343

Table 17: Comparative results for the multiple-disc clutch brake problem.

Algorithm	Optimal Values					Cost
	x_1	x_2	x_3	x_4	x_5	
SCA	71.238	91.238	1.0	772.881	2.125	0.25555
WOA	60.0	90.0	1.0	76.389	2.0	0.33081
QSA	70.0	90.0	1.0	1000.99	2.0	0.23524
NRO	70.144	90.171	1.0	881.899	2.012	0.23822
GBO	69.998	90.0	1.0	304.902	2.0	0.23526
AEO	70.0	90.0	1.0	869.392	2.0	0.23524
BMO	60.0	90.0	1.0	76.763	2.0	0.33081
GJO	69.995	90.0	1.0	130.745	2.0	0.23529
CDO	69.897	90.0	1.0	229.382	2.0	0.23634
NIO	70.0	90.0	1.0	239.093	2.0	0.23521

lights the effectiveness of **NIO** in navigating the complex search space of the problem and identifying a solution that outperforms those found by other established metaheuristic algorithms.

The optimal values identified by NIO for the design variables are $x_1 = 0.05$, $x_2 = 1.009$, $x_3 = 2.016$, and $x_4 = 500.0$, which collectively contribute to its superior performance. While algorithms like **QSA** approach **NIOs** cost (1.057), others, such as **WOA** (114.506), **AEO** (188.943), and **BMO** (413.945), struggle with constraints or convergence.

To sum up, the innovative mechanisms of **NIOs** make it a robust and versatile tool for addressing a wide range of complex engineering optimization problems. Its effectiveness has been demonstrated across multiple domains, including mechanical and structural engineering. From optimizing spring dimensions and gear configurations to improving structural integrity in truss and beam systems, **NIOs** consistently delivers competitive performance in balancing cost, efficiency, and mechanical constraints. Its success in solving high-impact problems, such as pressure vessels, clutch brake

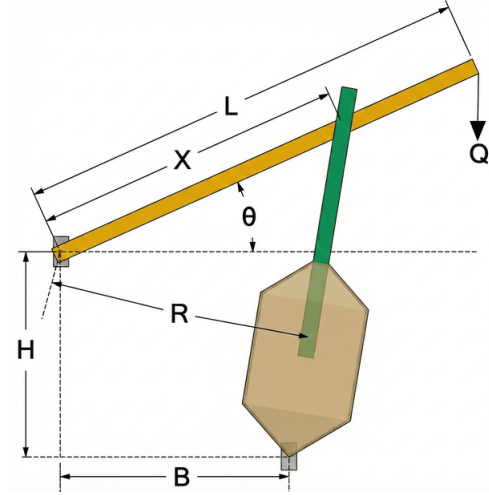


Figure 15: Piston lever design problem.

Table 18: Comparative results for the piston lever design problem.

Algorithm	Optimal Values				Cost
	x_1	x_2	x_3	x_4	
SCA	0.068	1.346	2.299	407.405	1.841
WOA	11.105	21.08	3.091	161.143	114.506
QSA	0.05	1.008	2.016	500.0	1.057
NRO	0.103	1.145	2.172	438.527	1.515
GBO	0.05	1.051	2.034	500.0	1.116
AEO	147.847	0.813	1.625	453.375	188.943
BMO	80.779	10.641	3.659	84.316	413.945
GJO	0.059	1.045	2.021	500.0	1.118
CDO	0.211	1.229	2.113	470.019	1.788
NIO	0.05	1.009	2.016	500.0	1.05

systems, and piston levers, emphasizes its adaptability and robustness. These results underscore the algorithm's potential not only for advancing engineering design practices but also for inspiring further research and development in metaheuristic optimization.

5 Conclusion

In this paper, a novel meta-heuristic optimization technique is introduced, the Nubian Ibex Optimizer, inspired by the adaptive and agile behaviors of the Nubian ibex in its natural habitat. The NIO algorithm mimics ibex's strategies for navigating challenging terrains, translating these into efficient search and optimization processes. To evaluate the effectiveness of the NIO, a comprehensive set of benchmark tests is employed, including twenty-three classical benchmark functions and a selection of eight complex engineering problems. These tests were designed to assess the NIO's

capability in both the exploration and exploitation phases of optimization.

The statistical results from these benchmarks indicate that the NIO algorithm demonstrates remarkable performance. Particularly noteworthy is the NIO's ability to efficiently navigate and find solutions in high-dimensional and complex landscapes, a trait analogous to the ibex's skill in traversing difficult terrains. It not only competes favorably with established meta-heuristic algorithms like SCA, WOA, QSA, NRO, GBO, AEO, BMO, GJO, and CDO but in several instances, surpasses them. In the domain of real-world applications, the NIO has shown promising results, especially in engineering problems, known for their high dimensionality and complexity. These applications demonstrate the NIO's adaptability and robustness in both constrained and unconstrained optimization scenarios.

Looking forward, the NIO algorithm opens up numerous possibilities for future research and application. Its potential applications span a wide range of fields, including, but not limited to, renewable energy optimization, neural network training, image and signal processing, data mining, resource management, network optimization, and industrial engineering challenges. Furthermore, the NIO can be adapted for multi-objective optimization problems, which are increasingly prevalent in complex system design and management. Additionally, the NIO offers opportunities for enhancement and hybridization. Integrating advanced strategies such as disruption techniques, mutation strategies, and other stochastic components could further enhance its exploration and exploitation capabilities. The development of a discrete version of the NIO could also extend its applicability to a broader range of optimization problems.

To conclude, the NIO algorithm, with its unique inspiration and robust performance across various benchmarks and real-world applications, represents a significant advancement in the field of meta-heuristic optimization algorithms. Its versatility and effectiveness suggest that it will be a valuable tool for solving complex optimization problems in numerous scientific and engineering domains.

Acknowledgment

The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP2/286/46 and Universiti Teknologi PETRONAS under the Yayasan Universiti Teknologi PETRONAS (YUTP FRG/015LC0-578) for providing the required facilities to conduct this research work.

Declaration of competing interest

The authors declare no financial or personal interests that could have influenced this work.

Data availability

The Python NIO codes are available on GitHub, <https://github.com/mogragab/nio>

Declaration of generative AI and AI-assisted technologies

”The authors used AI tools to enhance language and readability, then reviewed and edited the content to ensure accuracy and take full responsibility for the final publication.”

Glossary

- D Dimension of the problem. 7, 8, 15
- I Population initialization. 16
- LB Lower bound of the search space. 7, 8
- N Population size (number of ibexes). 7, 8, 14–16
- R Random number between 0 and 1. 8, 11, 16
- T Maximum number of iterations. 15, 16
- UB Upper bound of the search space. 7, 8
- X The potential solutions. 7
- X_i Position of the i^{th} ibex. 7
- X_{best} Global best position of the herd. 10, 11, 13
- X_{mean} Mean position of the herd. 10
- X_{rand} Randomly selected ibex position. 11
- α Agility parameter. 12, 13, 15, 16
- ω Distance to the point of interest. 9
- ψ Jump strength. 10, 11
- ρ Path’s radius of curvature. 9, 15, 16
- σ Scaling factor determined by the gamma function. 9, 10
- θ Movement angle. 9, 11
- ϑ Precision parameter. 12, 13, 15, 16
- K The total number of independent runs. 16, 17

ABC Artificial Bee Colony. [4](#)

ACO Ant Colony Optimization. [4](#)

AEO Artificial Ecosystem Optimization. [17](#), [28](#), [40](#)

AO Aquila Optimizer. [9](#)

BMO Barnacles Mating Optimizer. [5](#), [17](#), [28](#), [40](#)

BSO Brain Storm Optimization. [4](#)

CDO Chernobyl Disaster Optimization. [4](#), [17](#)

CPO Crested Porcupine Optimizer. [3](#)

EBO Ecogeography-Based Optimization. [5](#)

EVO Energy Valley Optimization. [4](#)

FA Firefly Algorithm. [4](#)

FHO Fire Hawk Optimizer. [3](#)

FIA Fibonacci Indicator Algorithm. [5](#)

FOA Forest Optimization Algorithm. [3](#)

GA Genetic Algorithm. [3](#)

GBO Gradient-Based Optimizer. [17](#), [35](#)

GCO Germinal Center Optimization. [4](#), [5](#)

GEA Gradient Evolution Algorithm. [5](#)

GJO Golden Jackal Optimization. [17](#)

GSA Gravitational Search Algorithm. [4](#)

GWO Grey Wolf Optimizer. [4](#)

HHO Harris Hawks Optimization. [9](#)

IMO Ions Motion Optimization. [3](#)

ITGO Invasive Tumor Growth Optimization. [5](#)

JA Jaguar Algorithm. [3](#)

LOA Lion Optimization Algorithm. [3](#)

MBF Mouth Breeding Fish Algorithm. [3](#)

ML Machine Learning. [2](#)

NIO Nubian Ibex Optimization. [3](#), [7](#), [8](#), [10–18](#), [20](#), [21](#), [28](#), [35](#), [39](#), [40](#)

NRO Nuclear Reaction Optimization. [17](#), [20](#)

OOA Osprey Optimization Algorithm. [4](#)

PBA Pity Beetle Algorithm. [5](#)

PRS Prism Refraction Search. [4](#)

PSO Particle Swarm Optimization. [4](#)

QS Queuing Search Algorithm. [4](#)

QSA Queuing Search Algorithm. [17](#), [40](#)

RIME Physical Phenomenon of RIME-ice. [4](#)

ROA Rain Optimization Algorithm. [3](#)

RTO Root Tree Optimization Algorithm. [5](#)

SAR Search And Rescue Optimization. [4](#)

SCA Sine Cosine Algorithm. [5](#), [17](#)

SGO Soccer Game Optimization. [4](#)

SOA Seagull Optimization Algorithm. [5](#)

SOS Symbiotic Organisms Search. [4](#)

SSO Shark Smell Algorithm. [5](#)

SSO Social Spider Optimization. [4](#)

TS Tabu Search. [4](#)

WAOA Walrus Optimization Algorithm. [9](#)

WCA Water Cycle Algorithm. [4](#), [5](#)

WGA Wild Goats Algorithm. [5](#), [10](#), [11](#), [13](#)

WO Walrus Optimizer. [4](#)

WOA Whale Optimization Algorithm. [17](#), [40](#)

References

- [1] Mohamed Abdel-Basset, Laila Abdel-Fatah, and Arun Kumar Sangaiah. “Metaheuristic algorithms: A comprehensive review”. In: *Computational intelligence for multimedia big data on the cloud with engineering applications* (2018), pp. 185–231.
- [2] Mohamed Abdel-Basset, Reda Mohamed, and Mohamed Abouhawwash. “Crested Porcupine Optimizer: A new nature-inspired metaheuristic”. In: *Knowledge-Based Systems* 284 (2024), p. 111257.
- [3] Mohamed Abdel-Basset and Laila A Shawky. “Flower pollination algorithm: a comprehensive review”. In: *Artificial Intelligence Review* 52 (2019), pp. 2533–2557.
- [4] Sina Abdipoor et al. “Meta-heuristic approaches for the University Course Timetabling Problem”. In: *Intelligent Systems with Applications* (2023), p. 200253.
- [5] Benyamin Abdollahzadeh, Farhad Soleimanian Gharehchopogh, and Seyedali Mirjalili. “African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems”. In: *Computers & Industrial Engineering* 158 (2021), p. 107408.
- [6] Laith Abualigah. “Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications”. In: *Neural Computing and Applications* 33.7 (2021), pp. 2949–2972.
- [7] Laith Abualigah et al. “Aquila optimizer: a novel meta-heuristic optimization algorithm”. In: *Computers & Industrial Engineering* 157 (2021), p. 107250.
- [8] Laith Abualigah et al. “Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results”. In: *Neural Computing and Applications* (2022), pp. 1–30.
- [9] Laith Abualigah et al. “The arithmetic optimization algorithm”. In: *Computer methods in applied mechanics and engineering* 376 (2021), p. 113609.
- [10] Iman Ahmadianfar, Omid Bozorg-Haddad, and Xuefeng Chu. “Gradient-based optimizer: A new metaheuristic optimization algorithm”. In: *Information Sciences* 540 (2020), pp. 131–159.
- [11] Bahriye Akay, Dervis Karaboga, and Rustu Akay. “A comprehensive survey on optimizing deep learning models by metaheuristics”. In: *Artificial Intelligence Review* 55.2 (2022), pp. 829–894.
- [12] Hamzeh Mohammad Alabool et al. “Harris hawks optimization: a comprehensive review of recent variants and applications”. In: *Neural computing and applications* 33 (2021), pp. 8939–8980.
- [13] Alireza Askarzadeh. “Bird mating optimizer: an optimization algorithm inspired by bird mating strategies”. In: *Communications in Nonlinear Science and Numerical Simulation* 19.4 (2014), pp. 1213–1228.
- [14] Salih Berkan Aydemir. “Ideal solution candidate search for starling murmuration optimizer and its applications on global optimization and engineering problems: SB Aydemir”. In: *The Journal of Supercomputing* 80.3 (2024), pp. 4083–4156.
- [15] Tummala SLV Ayyarao et al. “War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization”. In: *IEEE Access* 10 (2022), pp. 25073–25105.
- [16] Mahdi Azizi, Siamak Talatahari, and Amir H Gandomi. “Fire Hawk Optimizer: A novel metaheuristic algorithm”. In: *Artificial Intelligence Review* 56.1 (2023), pp. 287–363.
- [17] Mahdi Azizi et al. “Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization”. In: *Scientific Reports* 13.1 (2023), p. 226.
- [18] Thomas Bäck and Hans-Paul Schwefel. “An overview of evolutionary algorithms for parameter optimization”. In: *Evolutionary computation* 1.1 (1993), pp. 1–23.
- [19] Ashok D Belegundu and Jasbir S Arora. “A study of mathematical programming methods for structural optimization. Part I: Theory”. In: *International Journal for Numerical Methods in Engineering* 21.9 (1985), pp. 1583–1599.

- [20] Vivien J Chebii et al. “Genome scan for variable genes involved in environmental adaptations of Nubian ibex”. In: *Journal of Molecular Evolution* 89.7 (2021), pp. 448–457.
- [21] Vivien J Chebii et al. “Genome-wide analysis of Nubian ibex reveals candidate positively selected genes that contribute to its adaptation to the desert environment”. In: *Animals* 10.11 (2020), p. 2181.
- [22] Chin-Chi Chen et al. “A novel metaheuristic: Jaguar algorithm with learning behavior”. In: *2015 IEEE international conference on systems, man, and cybernetics*. IEEE. 2015, pp. 1595–1600.
- [23] Min-Yuan Cheng and Doddy Prayogo. “Symbiotic organisms search: a new metaheuristic optimization algorithm”. In: *Computers & Structures* 139 (2014), pp. 98–112.
- [24] Haruna Chiroma et al. “Nature inspired meta-heuristic algorithms for deep learning: recent progress and novel perspective”. In: *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1* 1. Springer. 2020, pp. 59–70.
- [25] Bastien Chopard et al. “Performance and limitations of metaheuristics”. In: *An Introduction to Metaheuristics for Optimization* (2018), pp. 191–203.
- [26] Nitish Chopra and Muhammad Mohsin Ansari. “Golden jackal optimization: A novel nature-inspired optimizer for engineering applications”. In: *Expert Systems with Applications* 198 (2022), p. 116924.
- [27] David W Corne and Michael A Lones. “Evolutionary algorithms”. In: *arXiv preprint arXiv:1805.11014* (2018).
- [28] Erik Cuevas et al. “A swarm optimization algorithm inspired in the behavior of the social-spider”. In: *Expert Systems with Applications* 40.16 (2013), pp. 6374–6384.
- [29] Ashraf Darwish. “Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications”. In: *Future Computing and Informatics Journal* 3.2 (2018), pp. 231–246.
- [30] Mohammad Dehghani, Eva Trojovská, and Tomáš Zušćák. “A new human-inspired metaheuristic algorithm for solving optimization problems based on mimicking sewing training”. In: *Scientific Reports* 12.1 (2022), p. 17387.
- [31] Mohammad Dehghani and Pavel Trojovský. “Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems”. In: *Frontiers in Mechanical Engineering* 8 (2023), p. 1126450.
- [32] Gaurav Dhiman and Vijay Kumar. “Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems”. In: *Knowledge-based systems* 165 (2019), pp. 169–196.
- [33] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. “Ant colony optimization”. In: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.
- [34] Jose A Egea et al. “MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics”. In: *BMC bioinformatics* 15 (2014), pp. 1–9.
- [35] MA Eita and MM Fahmy. “Group counseling optimization”. In: *Applied Soft Computing* 22 (2014), pp. 585–604.
- [36] Hadi Eskandar et al. “Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems”. In: *Computers & Structures* 110 (2012), pp. 151–166.
- [37] A Etmianiesfahani, Afshin Ghanbarzadeh, and Z Marashi. “Fibonacci indicator algorithm: A novel tool for complex optimization problems”. In: *Engineering Applications of Artificial Intelligence* 74 (2018), pp. 1–9.
- [38] Manizheh Ghaemi and Mohammad-Reza Feizi-Derakhshi. “Forest optimization algorithm”. In: *Expert Systems with Applications* 41.15 (2014), pp. 6676–6687.
- [39] Hadi Gholizadeh et al. “Modelling uncertainty in sustainable-green integrated reverse logistics network using metaheuristics optimization”. In: *Computers & Industrial Engineering* 163 (2022), p. 107828.
- [40] Fred Glover and Manuel Laguna. *Tabu search*. Springer, 1998.
- [41] Lei Han et al. “Different environmental requirements of female and male Siberian ibex, *Capra sibirica*”. In: *Scientific reports* 11.1 (2021), p. 6064.

- [42] Muxuan Han et al. “Walrus optimizer: A novel nature-inspired metaheuristic algorithm”. In: *Expert Systems with Applications* 239 (2024), p. 122413.
- [43] John H Holland. “Genetic algorithms”. In: *Scientific american* 267.1 (1992), pp. 66–73.
- [44] Essam H Houssein et al. “Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends”. In: *Swarm and Evolutionary Computation* 62 (2021), p. 100841.
- [45] Ehsan Jahani and Mohammad Chizari. “Tackling global optimization problems with a novel algorithm–Mouth Brooding Fish algorithm”. In: *Applied Soft Computing* 62 (2018), pp. 987–1002.
- [46] Zoran Jakšić et al. “A Comprehensive Review of Bio-Inspired Optimization Algorithms Including Applications in Microelectronics and Nanophotonics”. In: *Biomimetics* 8.3 (2023). issn: 2313-7673. doi: [10.3390/biomimetics8030278](https://doi.org/10.3390/biomimetics8030278).
- [47] JQ James and Victor OK Li. “A social spider algorithm for global optimization”. In: *Applied soft computing* 30 (2015), pp. 614–627.
- [48] Behzad Javidy, Abdolreza Hatamlou, and Seyedali Mirjalili. “Ions motion algorithm for solving optimization problems”. In: *Applied soft computing* 32 (2015), pp. 72–79.
- [49] Nikos Ath Kallioras, Nikos D Lagaros, and Dimitrios N Avtzis. “Pity beetle algorithm–A new metaheuristic inspired by the behavior of bark beetles”. In: *Advances in engineering software* 121 (2018), pp. 147–166.
- [50] Dervis Karaboga and Bahriye Akay. “A comparative study of artificial bee colony algorithm”. In: *Applied mathematics and computation* 214.1 (2009), pp. 108–132.
- [51] A Kaveh and M Khayatazad. “A new meta-heuristic method: ray optimization”. In: *Computers & structures* 112 (2012), pp. 283–294.
- [52] Ali Kaveh. *Applications of metaheuristic optimization algorithms in civil engineering*. Springer, 2017.
- [53] Mehrdad Kaveh and Mohammad Saadi Mesgari. “Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review”. In: *Neural Processing Letters* 55.4 (2023), pp. 4519–4622.
- [54] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.
- [55] Amel Ksibi, Anis Ben Ammar, and Chokri Ben Amar. “Enhancing relevance re-ranking using nature-inspired meta-heuristic optimization algorithms”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 1435–1442.
- [56] Rohit Kundu et al. “Prism refraction search: A novel physics-based metaheuristic algorithm”. In: *The Journal of Supercomputing* 80.8 (2024), pp. 10746–10795.
- [57] Ren-Jieh Kuo and Ferani E Zulvia. “The gradient evolution algorithm: A new metaheuristic”. In: *Information Sciences* 316 (2015), pp. 246–265.
- [58] Yacine Labbi et al. “A new rooted tree optimization algorithm for economic dispatch with valve-point effect”. In: *International Journal of Electrical Power & Energy Systems* 79 (2016), pp. 298–311.
- [59] Wenyi Long et al. “A constrained multi-objective optimization algorithm using an efficient global diversity strategy”. In: *Complex & Intelligent Systems* 9.2 (2023), pp. 1455–1478.
- [60] Seyedali Mirjalili. “SCA: a sine cosine algorithm for solving optimization problems”. In: *Knowledge-based systems* 96 (2016), pp. 120–133.
- [61] Seyedali Mirjalili and Andrew Lewis. “The whale optimization algorithm”. In: *Advances in engineering software* 95 (2016), pp. 51–67.
- [62] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. “Grey wolf optimizer”. In: *Advances in engineering software* 69 (2014), pp. 46–61.
- [63] Ali Reza Moazzeni and Ehsan Khamehchi. “Rain optimization algorithm (ROA): A new metaheuristic method for drilling optimization solutions”. In: *Journal of Petroleum Science and Engineering* 195 (2020), p. 107512.

- [64] Sahar Mohammad-Azari, Omid Bozorg-Haddad, and Xuefeng Chu. “Shark smell optimization (SSO) algorithm”. In: *Advanced optimization by nature-inspired algorithms* (2018), pp. 93–103.
- [65] Bernardo Morales-Castañeda et al. “An improved Simulated Annealing algorithm based on ancient metallurgy techniques”. In: *Applied Soft Computing* 84 (2019), p. 105761.
- [66] Mohammad H Nadimi-Shahraki et al. “MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems”. In: *Applied Soft Computing* 97 (2020), p. 106761.
- [67] Hindriyanto Dwi Purnomo and Hui-Ming Wee. “Soccer game optimization with substitute players”. In: *Journal of computational and applied mathematics* 283 (2015), pp. 79–90.
- [68] Hindriyanto Dwi Purnomo et al. “Soccer game optimization for travelling salesman problem”. In: *2017 International Conference on Innovative and Creative Information Technology (ICITech)*. Ieee. 2017, pp. 1–7.
- [69] Rebika Rai et al. “Human-inspired optimization algorithms: Theoretical foundations, algorithms, open-research issues and application for multi-level thresholding”. In: *Archives of Computational Methods in Engineering* 29.7 (2022), pp. 5313–5352.
- [70] Kanchan Rajwar, Kusum Deep, and Swagatam Das. “An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges”. In: *Artificial Intelligence Review* (2023), pp. 1–71.
- [71] Esmat Rashedi, Hossein Nezamabadi-Pour, and Saeid Saryazdi. “GSA: a gravitational search algorithm”. In: *Information sciences* 179.13 (2009), pp. 2232–2248.
- [72] Kumesan Reddy and Akshay K Saha. “A review of swarm-based metaheuristic optimization techniques and their application to doubly fed induction generator”. In: *Heliyon* 8.10 (2022).
- [73] Samira Sadeghi-Moghaddam, Mostafa Hajiaghahi-Keshteli, and Mehdi Mahmoodjanloo. “New approaches in metaheuristics to solve the fixed charge transportation problem in a fuzzy environment”. In: *Neural computing and applications* 31.Suppl 1 (2019), pp. 477–497.
- [74] Amir Shabani et al. “Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems”. In: *Expert Systems with Applications* 161 (2020), p. 113698.
- [75] Vivek Sharma and Ashish Kumar Tripathi. “A systematic review of meta-heuristic algorithms in IoT based application”. In: *Array* 14 (2022), p. 100164.
- [76] Alireza Shefaei and Behnam Mohammadi-Ivatloo. “Wild goats algorithm: An evolutionary algorithm to solve the real-world optimization problems”. In: *IEEE Transactions on Industrial Informatics* 14.7 (2017), pp. 2951–2961.
- [77] Hisham A Shehadeh. “Chernobyl disaster optimizer (CDO): a novel meta-heuristic method for global optimization”. In: *Neural Computing and Applications* 35.15 (2023), pp. 10733–10749.
- [78] Yuhui Shi. “Brain storm optimization algorithm”. In: *Advances in Swarm Intelligence: Second International Conference, ICSI 2011, Chongqing, China, June 12-15, 2011, Proceedings, Part I 2*. Springer. 2011, pp. 303–309.
- [79] Amparo Soler-Dominguez, Angel A Juan, and Renatas Kizys. “A survey on financial applications of metaheuristics”. In: *ACM Computing Surveys (CSUR)* 50.1 (2017), pp. 1–23.
- [80] Seshadri Sridharan, Ramesh Kumar Subramanian, and Arun Kumar Srirangan. “Physics based meta heuristics in manufacturing”. In: *Materials Today: Proceedings* 39 (2021), pp. 805–811.
- [81] Hang Su et al. “RIME: A physics-based optimization”. In: *Neurocomputing* 532 (2023), pp. 183–214.
- [82] Mohd Herwan Sulaiman et al. “Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems”. In: *Engineering Applications of Artificial Intelligence* 87 (2020), p. 103330.
- [83] Deyu Tang et al. “ITGO: Invasive tumor growth optimization algorithm”. In: *Applied Soft Computing* 36 (2015), pp. 670–698.

- [84] Qasem Al-Tashi et al. “A review of grey wolf optimizer-based feature selection methods for classification”. In: *Evolutionary machine learning techniques: algorithms and applications* (2020), pp. 273–286.
- [85] Eva Trojovská, Mohammad Dehghani, and Pavel Trojovský. “Fennec fox optimization: A new nature-inspired optimization algorithm”. In: *IEEE Access* 10 (2022), pp. 84417–84443.
- [86] Chun-Wei Tsai et al. “Metaheuristic algorithms for healthcare: Open issues and challenges”. In: *Computers & Electrical Engineering* 53 (2016), pp. 421–434.
- [87] Carlos Villaseñor et al. “Germinal center optimization algorithm”. In: *International Journal of Computational Intelligence Systems* 12.1 (2018), pp. 13–27.
- [88] Gai-Ge Wang, Suash Deb, and Leandro Dos Santos Coelho. “Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems”. In: *International journal of bio-inspired computation* 12.1 (2018), pp. 1–22.
- [89] Zhenglei Wei et al. “Nuclear reaction optimization: A novel and powerful physics-based algorithm for global optimization”. In: *IEEE Access* 7 (2019), pp. 66084–66109.
- [90] Pawan Whig et al. “Leveraging Meta-Heuristics in Improving Health Care Delivery: A Comprehensive Overview”. In: *Nature-Inspired Methods for Smart Healthcare Systems and Medical Data* (2023), pp. 149–168.
- [91] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.
- [92] Xin-She Yang. “Firefly algorithm, stochastic test functions and design optimisation”. In: *International journal of bio-inspired computation* 2.2 (2010), pp. 78–84.
- [93] Maziar Yazdani and Fariborz Jolai. “Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm”. In: *Journal of computational design and engineering* 3.1 (2016), pp. 24–36.
- [94] Jinhao Zhang et al. “Queueing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems”. In: *Applied Mathematical Modelling* 63 (2018), pp. 464–490.
- [95] Yudong Zhang, Shuihua Wang, Genlin Ji, et al. “A comprehensive survey on particle swarm optimization algorithm and its applications”. In: *Mathematical problems in engineering* 2015 (2015).
- [96] Weiguo Zhao, Liying Wang, and Zhenxing Zhang. “Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm”. In: *Neural Computing and Applications* 32 (2020), pp. 9383–9425.
- [97] Yu-Jun Zheng, Hai-Feng Ling, and Jin-Yun Xue. “Ecogeography-based optimization: enhancing biogeography-based optimization with ecogeographic barriers and differentiations”. In: *Computers & Operations Research* 50 (2014), pp. 115–127.
- [98] Yi Zhou, Weili Xia, and Jiapeng Dai. “The application of nature-inspired optimization algorithms on the modern management: A systematic literature review and bibliometric analysis”. In: *Journal of Management & Organization* 29.4 (2023), pp. 655–678.