Received: January 3, 2026. Revised: February 18, 2026.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Glass Lizard Optimization (GLO): A Novel Nature-inspired Metaheuristic Algorithm for Engineering Applications

Tareq Hamadneh[1] Omar Al Sayyed[2] Fadil A. Jaradat[3] Aseel Smerat[4] Mirjalol Ismoilov[5] Raed Abu Zitar[6] Zeinab Montazeri[7] Mohammad Dehghani[7]* Kei Eguchi[8]

[1]*Department of Mathematics, Al Zaytoonah University of Jordan, Amman 11733, Jordan*
[2]*Department of Mathematics, Faculty of Science, The Hashemite University, P.O. Box 330127, Zarqa 13133, Jordan*
[3]*Faculty of Information Technology, Abu Dhabi University, Abu Dhabi, United Arab Emirates*
[4]*Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan*
[5]*Department of Transport systems, Urgench State University named after Abu Rayhan Biruni, Urgench, 14, Kh.Alimdjan str, Urgench city, 220100, Uzbekistan*
[6]*College of Engineering and Computing, Liwa University, Abu Dhabi 41009, United Arab Emirates*
[7]*Department of Electrical and Electronics Engineering, Shiraz University of Technology, Shiraz 7155713876, Iran*
[8]*Department of Information Electronics, Fukuoka Institute of Technology, Japan*
* Corresponding author's Email: adanbax@gmail.com

Abstract: In this paper, a new metaheuristic algorithm called Glass Lizard Optimization (GLO) is introduced, inspired by the movement behaviors and hunting strategies of the glass lizard in nature. The design of GLO is based on two distinctive behaviors of this reptile: long-range creeping movements with spiral trajectories for extensive exploration and discovery of unknown regions, and short, focused, oscillatory movements for precise improvement of solutions around promising points. The initial population of the algorithm is generated randomly, with each member representing a candidate solution. In each iteration, the positions of the members are updated using dedicated equations from the exploration and exploitation phases, such that the exploration phase, through long-range and twisting movements, enables wide search coverage and prevents premature convergence, while the exploitation phase, with short and oscillatory movements, facilitates precise solution refinement and effective convergence. The performance of GLO was evaluated on 22 constrained optimization problems from the CEC 2011 benchmark suite and compared with nine well-known metaheuristic algorithms. The results show that GLO achieved the best mean solution in 19 out of 22 problems, demonstrating stable, competitive, and reliable performance. The Wilcoxon statistical analysis confirms the significant superiority of GLO over the competing methods. These findings indicate that, by intelligently combining the exploration and exploitation phases and leveraging the behavioral model of the lizard, GLO serves as an effective and dependable tool for solving complex constrained optimization problems in real-world applications.

Keywords: Glass lizard, Swarm-inspired, Engineering, Exploration, Exploitation, Optimization, Metaheuristic.

1. Introduction

Optimization refers to the process of finding the best solution from a set of feasible alternatives under specified constraints. This concept constitutes one of the fundamental pillars in engineering sciences, management, economics, and many applied domains, as real-world problems are often defined in such a way that obtaining an optimal solution is of vital importance [1]. Although classical mathematical methods have demonstrated acceptable performance in solving many optimization problems [2, 3], their efficiency becomes limited as problem structures grow more complex, dimensionality increases,

relationships become nonlinear, and multiple constraints are involved. Consequently, the need for more efficient tools has become increasingly evident [4].

Under such circumstances, metaheuristic algorithms have been introduced as one of the most effective stochastic search-based approaches. By employing guided search mechanisms within the solution space, these algorithms are capable of providing high-quality solutions for complex problems without requiring gradient information or restrictive mathematical assumptions. High flexibility, ease of implementation, and the ability to cope with multimodal search spaces have made metaheuristics widely used tools in solving optimization problems [5].

The search process in metaheuristic algorithms is based on two fundamental principles: exploration and exploitation [6]. Exploration, associated with global search, refers to broad and diverse scanning of different regions of the solution space with the aim of discovering promising areas and avoiding entrapment in local optima. In contrast, exploitation, associated with local search, focuses on intensifying the search around high-quality solutions and progressively improving them. In essence, exploration determines "where to search", while exploitation determines "how to improve." The success of a metaheuristic algorithm lies in maintaining a balanced management of these two processes throughout the search iterations, in such a way that neither search diversity is lost nor convergence accuracy is reduced [7].

Despite the high efficiency of these algorithms, the stochastic nature of the search process implies that there is no guarantee of reaching the global optimum. Algorithmic performance is inherently accompanied by uncertainty, and results may vary across different runs. Therefore, it cannot be claimed that a single metaheuristic algorithm delivers the best performance for all optimization problems. This concept is expressed through the well-known No Free Lunch (NFL) theorem [8], which states that the success of an algorithm on a set of problems does not guarantee similar performance on other problems. This perspective has provided a fundamental motivation for researchers to design and develop new metaheuristic algorithms with innovative search mechanisms.

Metaheuristic algorithms are generally classified into four main groups based on their primary source of inspiration. The first group includes swarm-based algorithms, developed by drawing inspiration from the collective behaviors and strategies of living organisms in nature; examples include Particle Swarm Optimization (PSO) [9], Falco peregrinus Optimization Algorithm (FOA) [10], Ant Colony Optimization (ACO) [11], Glider snake optimizer (GSO) [12], and Bolas Spider Algorithm (BSA) [13]. The second group includes evolutionary algorithms, rooted in genetic and biological concepts, employing mechanisms such as selection, mutation, and recombination in the search process; Genetic Algorithm (GA) [14] and Differential Evolution (DE) [15] are among the most well-known representatives of this category. The third group consists of physics-based algorithms, inspired by physical laws, forces, and phenomena; such as Gravitational Search Algorithm (GSA) [16], Henry Gas Solubility Optimization (HGSO) [17], Simulated Annealing (SA) [18], and Lichtenberg Algorithm (LA) [19]. Finally, human-behavior-based algorithms are designed according to human interactions, decision-making processes, and strategic behaviors; examples include Mother Optimization Algorithm (MOA) [20], Dance Training-Based Optimizer (DTBO) [21], Teaching–Learning-Based Optimization (TLBO) [22], Key Maker Algorithm (KMA) [23], and Detective Algorithm (DA) [24]. In addition to these four categories, numerous other algorithms have been introduced in the literature, each attempting to provide more effective search mechanisms.

Despite the wide diversity of existing algorithms, a review of the literature indicates that no metaheuristic algorithm has yet been designed with direct inspiration from the locomotion behaviors and movement strategies of the glass lizard reptile. This is while the search style of this reptile in nature exhibits distinctive behavioral characteristics. Long-range creeping displacements combined with spiral trajectories during environmental search enable extensive habitat coverage and facilitate the discovery of food resources. Conversely, when approaching prey in the final stage, this reptile shifts its strategy to short, concentrated, and oscillatory movements in order to stabilize the hunting position with high precision.

Accordingly, motivated by the concept of the NFL theorem as well as the intention to address this research gap, this study introduces and designs a new metaheuristic algorithm named Glass Lizard Optimization (GLO) to be applied to solving optimization problems across various domains.

The main contributions of this research can be summarized as follows. First, the GLO algorithm is introduced as a swarm-based metaheuristic approach inspired by the real behaviors of the glass lizard. Second, the theoretical and bio-behavioral concepts of the algorithm are explained and then mathematically modeled within a framework consisting of exploration and exploitation phases. In

the exploration phase, long-range creeping displacements and spiral paths are employed to simulate global search, whereas the exploitation phase is designed based on short, concentrated, and oscillatory movements to perform precise local search. Subsequently, the performance of the proposed algorithm in solving real-world constrained optimization problems is evaluated on 22 problems from the CEC 2011 test suite, and its results are compared with the performance of nine well-known metaheuristic algorithms.

The remainder of this paper is organized as follows. Section 2 presents the proposed GLO algorithm along with its complete mathematical modeling. Section 3 describes the simulation studies and performance evaluation of the algorithm in solving real-world optimization problems. Finally, Section 4 concludes the paper and provides suggestions for future research.

2. Glass lizard optimization (GLO)

This section is devoted to introducing and designing the proposed Glass Lizard Optimization (GLO) approach. First, the theoretical foundations and abstract concepts employed in shaping this algorithm are explained based on a precise and purposeful behavioural analysis of the glass lizard, so that the rationale behind selecting the search mechanisms becomes transparent and clearly understandable.

Subsequently, by extracting the reptile's representative locomotion patterns and conceptually mapping them onto optimization processes, the mathematical framework of the algorithm is developed in the form of two complementary phases: exploration and exploitation. Specifically, the exploration phase is modelled on long-range creeping displacements and spiral trajectories observed during environmental foraging, whereas the exploitation phase is formulated by drawing inspiration from the short, focused, and oscillatory movements exhibited when approaching prey.

In this way, the final structure of the algorithm emerges from a coherent integration of bio-behavioural concepts and mathematical modelling, providing a systematic mechanism for implementing both global and local search within the solution space.

2.1 Behavioural analysis and selection of representative glass lizard behaviours for designing the GLO algorithm

In the initial step of designing the Glass Lizard Optimization (GLO) algorithm, conducting a precise and purposeful behavioral analysis of the glass lizard's lifestyle and locomotion patterns is essential. The biological credibility of the algorithm can only be ensured when the mapping between real behavioral traits and the mathematical model is transparent, justifiable, and comprehensible. Owing to its elongated body structure, lack of well-developed locomotor limbs, and complete reliance on lateral undulatory motion, the glass lizard exhibits displacement patterns distinct from most reptiles. This reptile spends a significant portion of its time searching for food resources, locating shelter, and evaluating environmental conditions. Throughout this process, it employs a combination of extensive exploratory movements and focused, goal-directed movements. This behavioral duality forms the foundation for selecting two representative behaviors to model the exploration and exploitation phases in the GLO algorithm.

The first selected behavior, which forms the basis for designing the exploration phase, is the pattern of long-range creeping displacements accompanied by spiral trajectories during environmental search. While exploring its habitat, the glass lizard traverses relatively long distances through successive wave-like contractions and alternating changes in movement direction. Its path is neither linear nor straight; rather, it takes on a tortuous and deviational form, influenced by stimuli such as prey scent or environmental vibrations that induce random directional shifts. This locomotion pattern enables the reptile to cover a wide area instead of restricting itself to limited regions. From an optimization standpoint, such behavior directly corresponds to global search in the solution space, as the search agent—by distancing itself from its current position, moving toward random regions, and following nonlinear paths—can discover unknown areas and avoid premature convergence. At the same time, these displacements are not entirely blind; they are partially guided by favorable environmental cues. This concept is incorporated into the mathematical model through a guided tendency towards the best identified position. Therefore, selecting this behavior for the exploration phase is justified from two perspectives: first, broad coverage of the search space, and second, preservation of relative orientation towards promising regions.

In contrast, the behavior selected for designing the exploitation phase is the pattern of short, focused, and oscillatory displacements during the final approach to prey. When the glass lizard detects a food source at close range, it markedly alters its movement strategy. At this stage, long displacements are replaced by short, gradual, and fully controlled

motions. Through continuous adjustment of body angle, subtle locomotion oscillations, and progressive distance reduction, the reptile strives to stabilize its position with high precision. Although these movements have small amplitudes, they are highly purposeful and play a decisive role in hunting success. From an optimization perspective, this pattern is equivalent to an accurate local search around high-quality solutions, where the search agent focuses on gradual, fine-scale improvement rather than extensive exploration. The presence of subtle motion oscillations further enables the agent to perform delicate trajectory corrections without leaving promising regions, thereby reducing the risk of entrapment in local optima.

Accordingly, within the GLO algorithm framework, the long-range spiral creeping behavior has been selected as the governing mechanism of the exploration phase, as it exhibits the strongest conceptual correspondence with global search, path diversity, and discovery of unknown regions. Conversely, the short, focused, oscillatory displacement behavior has been adopted as the foundation of the exploitation phase, since it reflects gradual concentration, reduction in movement amplitude, and precise solution refinement around the best identified positions.

Overall, the selection of these two representative behaviors is not only fully defensible from a bio-behavioral standpoint, but also enables a clear mathematical mapping to global and local search operators. In the following, the algorithm initialization process is first described, after which the mathematical modeling of the aforementioned real behaviors for designing the exploration and exploitation phases within the GLO algorithm structure will be presented.

2.2 Initialization of the Glass Lizard Optimization (GLO) Algorithm

The first step in designing and mathematically modeling the Glass Lizard Optimization (GLO) algorithm is the population initialization process. GLO is a population-based algorithm in which each member, inspired by the exploratory behavior of the glass lizard, can effectively search the solution space and identify high-quality solutions. In this algorithm, each glass lizard corresponds to a population member, and its natural habitat represents the solution space. Therefore, each member of the population is considered a candidate solution for the optimization problem.

The position of each population member in the solution space is represented by a vector X_i, where each component of the vector corresponds to a decision variable of the problem. Consequently, the entire population can be represented as a population matrix in the following form:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \quad (1)$$

where:

- N is the number of population members
- m is the number of decision variables,
- $x_{i,d}$ is the value of the d-th decision variable for the i-th member of the population.

For each member, the initial position is generated randomly to ensure sufficient diversity within the population:

$$x_{i,d} = lb_d + r \cdot (ub_d - lb_d) \quad (2)$$

where lb_d and ub_d are the lower and upper bounds of the d-th decision variable, and ($r \in [0,1]$) is a uniformly distributed random number. Once the population members' positions are generated, the objective function of the problem is evaluated for each member to measure the quality of each candidate solution. The set of objective function values for the population is represented as:

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (3)$$

where $F_i = F(X_i)$ denotes the objective function value of the i-th population member.

After generating the initial positions and evaluating the objective function, the initialization phase is complete, and the population is ready to enter the iterative position-update process. In GLO, population positions are updated through two independent phases: Exploration and Exploitation, each modeled based on the representative behaviors of the glass lizard in nature. These two phases are subsequently described and mathematically formulated in detail.

2.3 Phase 1: Design of the exploration phase based on the long-range creeping behaviour of the glass lizard

In the exploration phase of the proposed algorithm, the selected representative behavior is derived from the glass lizard's lifestyle, particularly its long-range creeping displacements and spiral locomotion paths in its natural habitat. Due to its elongated body structure and the absence of well-developed locomotor limbs, this reptile moves through lateral undulatory contractions. When searching for food resources, locating shelter, or assessing environmental conditions, it tends to traverse relatively long distances while repeatedly changing its direction of movement, thereby exploring a wide portion of its habitat. Its trajectory is typically nonlinear and tortuous, and random deviations arise under the influence of environmental stimuli such as prey scent or ground vibrations. From an optimization perspective, this behavioral pattern corresponds to global search in the solution space, because the search agent—through wide displacements, path deviations, and distancing from its current position—can discover unexplored regions of the search space and avoid premature convergence. Nevertheless, even within this extensive search, the movement orientation is not entirely random and is partially influenced by favorable environmental cues; a concept that is mathematically modeled by incorporating a tendency towards the best identified position. Accordingly, the position of each glass lizard in the search space is represented by the vector (X_i), and its updated position after applying the exploration operator is denoted by (X_i^{P1}). The position update equation in this phase is defined as follows:

$$X_i^{P1} = X_i + r_1 \cdot (X_{Best} - I_1 \cdot X_i) + r_2 \cdot (X_R - X_i) + S_i \quad (4)$$

where:

- X_i: Current position of agent i
- X_i^{P1}: New position after the exploration phase
- X_{Best}: Best position identified in the population
- X_R : A randomly selected position from the population or search space
- $r_i \in [0,1]$: Random coefficients controlling movement step lengths
- $I_i \in \{1,2\}$: Intensity coefficient of the current position's influence in the tendency towards the best solution
- S_i: Spiral creeping movement component

In this equation, the term ($X_{Best} - I_1 \cdot X_i$) represents the guided tendency towards the best-known region, where the coefficient (I_1) regulates the dependency intensity on the current position. This concept corresponds to the glass lizard following favorable cues (such as prey scent or suitable thermal conditions) while still maintaining a certain distance from its previous location. The term ($X_R - X_i$) expresses long-range random displacement towards unknown regions, playing a primary role in exploring the solution space. Finally, the component (S_i) models the torsional creeping path, ensuring that the agent's movement is not linear but wave-like and spiral. To accurately model this locomotion torsion, the spiral component is defined as:

$$S_i = r_3 \cdot \sin(r_4 \cdot 2\pi) \cdot \frac{|X_{best} - X_R|}{t} \quad (5)$$

where:

- $r_3 \in [0,1]$: Random coefficient controlling torsion amplitude
- $r_4 \in [0,1]$: Random variable determining deviation angle
- $\sin(r_4 \cdot 2\pi)$ Oscillatory component induced by the body's wave motion
- $|X_{best} - X_R|$: Distance between the best and random positions (scale of spiral motion)
- t : Scaling parameter controlling movement intensity (e.g., search progress index)

Using the distance ($|X_{best} - X_R|$) indicates that the torsion intensity depends on the extent of the region being explored; the greater the distance between favorable and random regions, the larger the spiral motion amplitude. Dividing by (t) shows that as the search process progresses, deviation intensity decreases and movement gradually stabilizes—a concept consistent with the glass lizard's increasing environmental familiarity. After generating the new position, a survival selection mechanism is applied to preserve solution quality, defined as:

$$X_i = \begin{cases} X_i^{P1}, & F_i^{P1} < F_i \\ X_i, & else \end{cases} \quad (6)$$

where:

- F_i : Objective function value at the current position
- F_i^{P1} : Objective function value at the new exploratory position

According to this rule, if the new position yields better quality than the previous one, it replaces it; otherwise, the agent remains at its former position. This mechanism corresponds to behavioral survival in nature, as the glass lizard continues along a new path only if its conditions are more favorable than the previous location.

2.4 Phase 2: Design of the exploitation phase based on the focused and oscillatory displacements of the glass lizard

In designing the exploitation phase of the proposed algorithm, the selected representative behavior is inspired by the movement pattern of the glass lizard when it performs its final focus on prey. When this reptile detects a food source at close range, instead of making long displacements, it relies on short, gradual, and fully controlled movements. These motions are accompanied by continuous adjustments in body angle and subtle locomotion oscillations, allowing the reptile to reduce its distance to the target in a stepwise manner. From an optimization perspective, such behavior represents a precise and concentrated local search around promising regions of the solution space, where the search agent attempts to improve solution quality and converge to the optimum through small yet purposeful steps. Accordingly, the mathematical modeling of this phase has been formulated to preserve local focus while also providing a mechanism for fine trajectory adjustment. The position update relation in the exploitation phase is defined as follows:

$$X_i^{P2} = X_i + r_5 \cdot (X_{best} - I_2 \cdot X_i) \cdot e^{-\frac{t}{T}} + r_6 \cdot \sin(r_7 \cdot 2\pi) \cdot \frac{(X_{mean} - I_3 \cdot X_i)}{t} \quad (7)$$

In this equation, X_i denotes the current position of agent (i), and X_i^{P2} represents the newly proposed position in the exploitation phase. The vector X_{best} indicates the best solution obtained so far and serves as the focal point of movement. The random coefficient $r_i \in [0,1]$ regulates the movement intensity towards this superior solution, while the binary parameter $I_i \in \{1,2\}$ introduces directional diversity, preventing premature convergence.

The exponential term $(e^{-\frac{t}{T}})$ models an adaptive reduction mechanism in displacement amplitude. As the iteration number (t) increases, the movement step size gradually decreases. This feature enables the algorithm to adopt highly focused and precise behavior in later stages—conceptually analogous to the reptile's cautious approach towards prey.

In the second part of the equation, the population tendency effect is incorporated through the mean vector (X_{mean}). This component reflects the overall aggregation direction of high-quality solutions within the population. The coefficient $r_i \in [0,1]$ controls the intensity of this effect, and the binary parameter $I_i \in \{1,2\}$ enhances structural diversity in movement. Dividing by (t) ensures that, over time, the influence of collective tendency diminishes and the agent's focus progressively shifts towards the best solution.

The presence of the oscillatory term $(\sin(r_7 \cdot 2\pi))$ constitutes one of the key elements of the model. The random parameter $(r_i \in [0,1])$ generates bounded sinusoidal fluctuations. These oscillations correspond to the subtle, corrective body movements of the glass lizard during positional stabilization—motions that enable extremely fine path adjustments without large displacements. From a computational standpoint, this component facilitates micro-scale exploration around promising solutions and reduces the likelihood of entrapment in local optima.

After generating the candidate position, the selection strategy is applied as follows:

$$X_i = \begin{cases} X_i^{P2}, & F_i^{P2} < F_i \\ X_i, & else \end{cases} \quad (8)$$

where F_i^{P2} is the objective function value at the new position and F_i is the previous value. This greedy selection mechanism ensures that only improving displacements are accepted; otherwise, the previous position is retained.

Overall, the mathematical structure of this phase—through the integration of directed movement toward the best solution, adaptive reduction of displacement amplitude, population tendency influence, and sinusoidal corrective oscillations—provides a coherent framework for implementing precise local search.

2.5 Implementation process, steps, and operational mechanism of the glass lizard optimization (GLO) algorithm

The proposed Glass Lizard Optimization (GLO) approach is a population-based algorithm developed to solve optimization problems across various scientific and engineering domains by drawing inspiration from the locomotion patterns and search strategies of the glass lizard. The operational structure of this algorithm is formed upon the simultaneous simulation of global and local search,

employing a combination of two complementary phases—exploration and exploitation—to achieve high-quality solutions.

The execution procedure of the algorithm begins with the population initialization stage. In this stage, the initial positions of the population members within the solution space are generated completely at random using Eqs. (1) to (3) to ensure the initial diversity of solutions. Each population member represents a candidate solution, and the objective function value is computed for all members. After the initial evaluation, by comparing the objective function values, the best population member is identified and stored in the algorithm's memory as the initial best position.

Upon completion of initialization, the algorithm enters the iterative search process. At the beginning of each iteration, the positions of the population members are updated based on behavioral mechanisms derived from the glass lizard. This updating procedure is carried out through two separate yet complementary phases.

In the exploration phase, the positions of the search agents are updated based on long-range creeping displacements and spiral trajectories. Eqs. (4) to (6) constitute the mathematical framework of this phase. At this stage, each agent moves away from its current position, travels towards random regions, and applies spiral deviations, enabling wide exploration of the solution space and discovery of unknown regions. Meanwhile, a guided tendency towards the best identified position is also maintained so that the search does not become entirely blind or directionless. After generating the new position, the objective function value is recalculated, and through a survival selection mechanism, improved positions replace the previous ones.

Subsequently, the algorithm proceeds to the exploitation phase, modeled by Eqs. (7) and (8). In this phase, the movement strategy of the agents shifts from wide displacements to short, focused, and oscillatory motions. Each agent moves gradually towards the best position while simultaneously applying highly precise adjustments to its path under the influence of collective tendency and regulatory oscillations. The adaptive reduction of displacement amplitude over iterations causes the search process to become increasingly concentrated, allowing convergence to occur with higher precision. Similar to the exploration phase, only those positions that lead to an improvement in the objective function value are accepted.

After completing both phases, the new objective function values of all population members are evaluated, and the best member of the population is again identified and updated if necessary. In this manner, one full iteration of the algorithm is completed. This iteration-based process continues until the stopping criterion—typically the maximum number of iterations—is satisfied. Throughout this cycle, the dynamic interaction between global search (exploration) and local search (exploitation) establishes a balance between diversity and intensification within the solution space.

At the end of the algorithm's execution, the best position obtained over all iterations is presented as the final solution to the problem. The step-by-step implementation structure of this process is provided in the form of pseudocode in Algorithm 1, enabling the operational procedure of the method to be clearly and practically demonstrated.

Algorithm 1: Glass Lizard Optimization (GLO)

Input:
Objective function
Population size N
Number of decision variables m
Lower and upper bounds (LB, UB)
Maximum number of iterations T

Output:
Best solution X_{best}

Initialize population X_i randomly using Eq. (2)
Evaluate fitness $F_i = F(X_i),\ i = 1{:}N.$
Determine the best solution X_{best}

For t = 1 to T do
  Compute population mean position X_{mean}
  For each search agent i = 1 to N do
    % Exploration Phase %
    Generate spiral component S_i using Eq. (5)
    Calculate new position X_i^{P1} using Eq. (4):
      Evaluate $F_i^{P1} = F(X_i^{P1})$
      Update X_i using Eq. (6)
    % Exploitation Phase %
    Calculate new position X_i^{P2} using Eq. (7):
    Evaluate $F_i^{P2} = F(X_i^{P2})$
    Update X_i using Eq. (8)
  End For search agent i = 1 to N do
  Update global best solution X_{best}
End For t = 1 to T do

Return X_{best} as the optimal solution

2.6 Computational complexity evaluation of the glass lizard optimization (GLO) algorithm

In this subsection, the computational complexity of the proposed Glass Lizard Optimization (GLO) algorithm is examined in order to analyze its processing cost and scalability when dealing with optimization problems of varying dimensionalities. Evaluating computational complexity not only provides a clear view of the algorithm's time efficiency, but also enables structural comparison with other population-based algorithms. Considering the designed structure of GLO, the computational process of the algorithm can be divided into two main parts: the population initialization process and the iterative position-updating process.

In the first part, namely the initialization stage, a population consisting of N members is generated for a problem with m decision variables. As described in the initialization subsection, the position of each population member is defined as an m-dimensional vector, and the initial values of the vector components are assigned using uniformly distributed random numbers within the variable bounds. Therefore, generating the initial positions for the entire population requires the assignment of $N \times m$ components. Subsequently, the objective function is evaluated for all members, which is also dependent on the problem dimension. Hence, the computational complexity of the initialization stage can be expressed as:

$$O_{init} = O(N \cdot m)$$

The second part, which constitutes the core computational burden of the algorithm, is the iterative population updating process during the algorithm's execution. This process is carried out over T iterations, and in each iteration, the positions of all population members are updated in two independent yet complementary phases: exploration and exploitation.

In each of these phases, for every population member, vector operations are performed, including addition, subtraction, multiplication by random coefficients, application of nonlinear functions such as sine and exponential, as well as re-evaluation of the objective function. Since all these computations are performed on m -dimensional vectors, the computational cost of updating each member in each phase is $O(m)$.

Considering N population members and T algorithm iterations, the complexity of each of the exploration and exploitation phases can be expressed as:

$$O_{exploration} = O(N \cdot m \cdot T)$$

$$O_{exploitation} = O(N \cdot m \cdot T)$$

Since these two phases are executed for the entire population in every iteration, the computational complexity of the total updating process equals the sum of the costs of these two phases. Consequently, the overall complexity of the GLO algorithm is obtained as:

$$O_{total} = O(N \cdot m) + 2 \cdot O(N \cdot m \cdot T)$$

By factoring, it can also be represented in the more compact form:

$$O_{total} = O((N \cdot m)(1 + 2T))$$

This relationship indicates that the computational complexity of the algorithm grows linearly with the population size N, the problem dimension m, and the number of iterations T. Such a structure shows that, in terms of time order, GLO falls within the class of population-based metaheuristic algorithms with linear polynomial complexity and possesses scalability suitable for high-dimensional problems. Moreover, the absence of extremely expensive computational operators or complex reconfiguration mechanisms helps maintain the processing cost of the algorithm at a balanced level.

Overall, the presented analysis demonstrates that the computational structure of GLO, while benefiting from two complementary search phases, maintains an acceptable time complexity and establishes an appropriate balance between search accuracy and computational cost—an important characteristic for application in large-scale engineering optimization problems.

3. Simulation studies and performance analysis of GLO in real-world applications

This section is devoted to simulation studies and performance analysis of the proposed Glass Lizard Optimization (GLO) approach in addressing real-world applications. The primary objective of this section is to evaluate the algorithm's capability in solving constrained optimization problems and to assess its efficiency in comparison with other well-established metaheuristic methods. To this end, the performance of GLO has been examined on 22 constrained optimization problems from the standard CEC 2011 test suite, and its results have been compared with several well-known algorithms in

order to provide a clear picture of the algorithm's accuracy, stability, and competitive strength.

3.1 Introduction of the test suite, competitor algorithms, and implementation conditions

To enable a precise analysis of GLO's performance, the CEC 2011 benchmark suite has been selected. This suite comprises 22 constrained optimization problems derived from real-world engineering and industrial applications and, due to the complexity of their constraint structures, the ruggedness of their search spaces, and the presence of limited feasible regions, it is regarded as one of the authoritative references for evaluating optimization algorithms. Complete information regarding the mathematical specifications, constraint types, and detailed descriptions of each problem has been reported in [25].

To assess performance quality, the results of the proposed algorithm have been compared with nine competitor algorithms, including Tunicate Search Algorithm (TSA) [26], Reptile Search Algorithm (RSA) [27], Whale Optimization Algorithm (WOA) [28], Multi-Verse Optimizer (MVO) [29], Grey Wolf Optimizer (GWO) [30], Teaching–Learning-Based Optimization (TLBO) [22], Gravitational Search Algorithm (GSA) [16], Particle Swarm Optimization (PSO) [9], and Genetic Algorithm (GA) [14]. These algorithms are among the well-established and widely used methods for solving constrained problems, and their selection enables a comprehensive and fair comparison.

All algorithms have been implemented under identical experimental conditions. Each algorithm has been executed in 20 independent runs across all problems, with each independent run consisting of 1000 iterations. To provide an accurate statistical evaluation, the results have been reported using six statistical indicators: mean, best value, worst value, standard deviation, median, and rank. The primary criterion for ranking the algorithms in each problem has been the mean indicator, as it provides a realistic representation of the algorithm's overall performance across repeated runs. The complete results are presented in Table 1, and the boxplot diagrams derived from the algorithms' performance are illustrated in Fig. 1.

3.2 Results and performance analysis

The results obtained from implementing GLO and the competitor algorithms on the CEC 2011 suite are presented in the results table and the boxplot diagrams. An examination of the data indicates that the proposed algorithm demonstrates highly powerful and competitive performance. GLO has achieved the best mean value in the majority of the problems and has secured the first rank. Specifically, the algorithm ranked first in 19 out of the total 22 problems and failed to obtain the top rank in only three cases. This level of success—equivalent to approximately 86% of all problems—reflects the algorithm's strong capability in handling diverse constraint structures and its effective adaptability to complex search spaces.

Even in the problems where the first rank was not achieved, the performance gap between GLO and the best-performing method was minimal, and the algorithm still remained among the top performers. This observation indicates that the proposed approach maintains its competitive performance not only under favorable conditions but also in challenging scenarios.

From the perspective of solution quality, the mean and best values obtained by GLO show a noticeable margin compared to its competitors in most problems. In some functions, even the worst solution recorded by GLO has been reported to be better than the mean results of certain other algorithms. This behavior suggests that the algorithm's search mechanism is highly robust and does not suffer severe performance degradation even under unfavorable conditions.

The analysis of standard deviation further confirms the algorithm's high stability. In a large number of problems, the standard deviation is very low and, in some cases, close to zero. This characteristic indicates that the algorithm's performance is not sensitive to initial population settings and exhibits consistent behavior across independent runs. In other words, GLO not only produces high-quality solutions but also preserves this quality in a repeatable manner.

Overall ranking indicators also confirm the algorithm's superiority. GLO has achieved the lowest total rank and the best average rank among all methods, with its mean rank reported to be very close to the ideal value of 1. Furthermore, the results of the Wilcoxon statistical test reveal that the performance differences between GLO and its competitors are statistically significant, indicating that the observed superiority is not due to random chance.

The boxplot diagrams also provide clear visual evidence. The high compactness of the boxes, the short whiskers, and the placement of the median lines at better values all indicate low variability, high stability, and a concentration of results in optimal regions. Even in the problems where the first rank was not achieved, the structure of the plots still reflects the algorithm's strong competitiveness.

Table 1. Optimization results of CEC 2011 test functions

		GLO	GA	PSO	GSA	TLBO	MVO	WOA	WSO	RSA	TSA
C11-F1	mean	11.76355	25.26184	25.23526	24.30174	20.24704	15.57308	20.32134	23.05399	26.00412	19.95888
	best	11.52382	24.02604	23.71999	22.76643	19.50651	11.22652	16.71223	19.52319	25.15683	11.64243
	worst	12.00327	26.30087	26.77231	25.30651	21.04358	21.94822	23.671	24.87764	27.78769	25.1625
	std	0.855705	3.419496	4.323606	3.688023	2.414368	14.3719	9.064909	7.482811	3.738887	18.08216
	median	11.76355	25.36022	25.22436	24.56702	20.21904	14.5588	20.45108	23.90757	25.53597	21.51529
	rank	1	9	8	7	4	2	5	6	10	3
C11-F2	mean	-23.8311	-10.9435	-22.9389	-8.33884	-9.4618	-8.87913	-12.1863	-9.03112	-8.06028	-8.54571
	best	-24.9733	-12.6439	-24.3552	-12.4193	-11.5728	-12.3083	-18.0416	-11.2341	-8.64276	-15.296
	worst	-22.6192	-9.64137	-21.2469	-6.579	-7.7271	-4.09456	-8.73417	-7.29832	-7.42364	-5.75153
	std	4.080869	4.409458	4.800911	8.46626	4.988398	10.73646	12.93404	5.157671	1.701974	14.00538
	median	-23.866	-10.7444	-23.0766	-7.17854	-9.27364	-9.55682	-10.9847	-8.79602	-8.08736	-6.56768
	rank	1	4	2	9	5	7	3	6	10	8
C11-F3	mean	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
	best	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
	worst	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
	std	6.25E-20	1.2E-15	1.84E-18	1.1E-18	3.05E-14	1.1E-11	5.22E-18	8.98E-11	1.8E-10	1.74E-13
	median	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
	rank	1	5	2	3	6	8	4	9	10	7
C11-F4	mean	25.80752	87.06162	81.28225	61.15306	107.6317	29.09525	45.51206	54.43426	68.96566	36.16339
	best	24.13017	53.97978	62.61475	44.86864	103.2559	25.39551	40.07621	42.39687	49.41382	32.24906
	worst	27.27719	109.9217	96.51984	74.95144	113.2115	32.9791	49.92628	60.41169	78.11429	39.62322
	std	4.002131	75.4328	47.72697	44.59199	12.79727	11.50672	15.3292	25.40989	40.70313	10.08637
	median	25.91136	92.17251	82.99719	62.39609	107.0297	29.00319	46.02287	57.46424	74.16727	36.39065
	rank	1	9	8	6	10	2	4	5	7	3
C11-F5	mean	-30.8062	-9.38283	-6.08439	-23.9453	-15.6291	-29.5167	-23.917	-20.7097	-16.7272	-25.2091
	best	-34.087	-11.3821	-8.18285	-29.7644	-18.0871	-34.1637	-26.932	-22.36	-19.341	-28.4921
	worst	-28.4205	-7.71442	-1.77903	-17.6051	-9.17349	-26.1825	-20.8098	-18.6028	-13.8236	-21.0152
	std	7.992427	4.765667	9.143615	16.36083	13.3355	12.27992	9.793511	4.853038	8.806974	11.12922
	median	-30.3586	-9.2174	-7.18783	-24.2058	-17.628	-28.8602	-23.9631	-20.9381	-16.8722	-25.6645
	rank	1	9	10	4	8	2	5	6	7	3
C11-F6	mean	-18.38	0	0	-13.6587	0	-9.52344	-16.1337	-10.2028	-10.5136	-5.73084
	best	-19.5104	0	0	-19.5118	0	-23.0058	-18.5818	-15.6595	-11.5854	-22.9234
	worst	-16.8457	0	0	-8.9567	0	0	-14.9492	-7.94102	-9.51564	0
	std	3.43929	0	0	14.39673	0	35.43261	5.214902	11.29208	3.565426	35.43172
	median	-18.582	0	0	-13.0832	0	-7.54395	-15.5019	-8.60535	-10.4768	0
	rank	1	8	8	3	8	6	2	5	4	7
C11-F7	mean	0.972093	2.016904	1.232857	1.001877	2.055154	1.178508	2.016556	1.931276	2.155071	1.50143
	best	0.804282	1.896229	1.122736	0.929962	1.941489	0.906877	1.942498	1.738667	1.83069	1.343752
	worst	1.108949	2.120896	1.332044	1.119663	2.281374	1.567404	2.100984	2.043854	2.345592	1.679321
	std	0.430375	0.311701	0.320715	0.256866	0.481034	0.865777	0.206539	0.449499	0.701993	0.473983
	median	0.987571	2.025246	1.238325	0.978942	1.998877	1.119875	2.011372	1.971291	2.222001	1.491324
	rank	1	8	4	2	9	3	7	6	10	5
C11-F8	mean	220	225.5	301.5472	263.8854	236.25	227.75	289	301.125	349.25	229
	best	220	220	220	259	231	220	261	281.3	299	220
	worst	220	231	445.189	278	238	251	331	339.6	404	238
	std	0	19.63252	330.5992	29.09124	10.81962	47.91546	104.771	80.8975	133.1393	32.12594
	median	220	225.5	270.5	259.2707	238	220	282	291.8	347	229
	rank	1	2	9	6	5	3	7	8	10	4
C11-F9	mean	234434.8	2203698	1177387	976875.8	499173.3	89475.45	480398.7	767029.3	1301689	67016.97
	best	151173.1	2121145	926562.5	797800.9	423823.5	70694.94	334149	547482.8	848186.4	47686.78
	worst	303488.4	2282718	1335389	1075122	566732	118073.3	636710.5	875656.5	1528229	91240.32
	std	197253.1	204612.7	569498.2	378552.4	226873.7	65867.83	382448.2	459658.3	953052.8	58043.64
	median	241538.9	2205465	1223799	1017290	503068.8	84566.76	475367.7	822488.9	1415170	64570.4
	rank	3	10	8	7	5	2	4	6	9	1
C11-F10	mean	-13.2245	-10.0121	-10.3502	-11.1459	-10.2472	-11.5175	-9.78057	-10.94	-10.1972	-12.951
	best	-16.3541	-10.1232	-10.3786	-11.6637	-10.3152	-11.9708	-10.2871	-12.1212	-10.5958	-15.8864
	worst	-11.638	-9.93204	-10.2991	-10.3415	-10.1587	-10.6949	-9.56422	-10.2661	-9.81793	-11.4333
	std	6.560739	0.251873	0.112196	1.791573	0.222173	1.740206	1.049117	2.510379	0.9826	6.163255
	median	-12.4531	-9.99656	-10.3616	-11.2891	-10.2574	-11.7021	-9.63547	-10.6863	-10.1875	-12.2421
	rank	1	9	6	4	7	3	10	5	8	2
C11-F11	mean	570775.6	6718148	5689705	1698257	5689705	1054518	1406553	7976499	10733179	7761995
	best	428270.8	6683461	5689705	1620870	5689705	274068.4	1328342	7450770	10440242	6521542
	worst	845850.9	6742348	5689705	1816028	5689705	1886647	1555782	8880540	10907934	10538397
	std	585423.6	76746.46	0	256953.1	0	2043153	313775.5	1938878	629586.5	5776808
	median	504490.4	6723391	5689705	1678065	5689705	1028678	1371044	7787343	10792271	6994020
	rank	1	6	5	4	5	2	3	8	9	7
C11-F12	mean	2107544	16099954	2743614	6841368	15932193	1315342	6445335	10976841	15954845	5701172
	best	1972705	15530301	2467455	6651129	15349785	1150355	6219838	10479905	14807307	5370862
	worst	2234800	16543229	3053141	6935254	16718963	1471486	6728315	11446873	16959124	5903153

	std	401823.2	1414519	921008.9	409180.9	1785605	405774.5	706623.1	1268870	2728649	737037.2
	median	2111335	16163143	2726930	6889544	15830011	1319763	6416593	10990293	16026475	5765337
	rank	2	10	3	6	8	1	5	7	9	4
C11-F13	mean	15459.91	15998.17	15525.55	161659	16586.41	15467.35	15506.01	16012.44	16506.66	15526.36
	best	15447.08	15507.99	15497.28	89617.12	16031.28	15459.29	15486.06	15751.65	15996.18	15497.49
	worst	15481.18	16536.89	15552.04	258429.1	18117.19	15470.55	15536.22	16647.11	17774.94	15565.12
	std	47.56416	1530.468	79.63051	221253.3	3157.708	16.65762	73.6177	1316.179	2635.274	87.23923
	median	15455.69	15973.9	15526.44	149294.9	16098.59	15469.77	15500.88	15825.5	16127.77	15521.41
	rank	1	6	4	10	9	2	3	7	8	5
C11-F14	mean	18866.6	19192.45	19240.19	19188.73	99130.93	19303.16	19304.54	148193.7	276941.8	19539.52
	best	18623.94	18825.08	18883.63	18967.91	34240.75	19207.26	19124.94	111226.2	202857.5	19421.08
	worst	19148.63	19384.42	19407.34	19523.2	165882.9	19368.67	19453.53	210188.8	400961	19689.29
	std	752.1742	776.5718	764.5494	760.4611	183841.9	228.234	475.8463	137095.3	274307.2	344.82
	median	18846.91	19280.15	19334.9	19131.92	98200.03	19318.36	19319.85	135679.9	251974.4	19523.85
	rank	1	3	4	2	8	5	6	9	10	7
C11-F15	mean	33013.63	7663278	33331.78	335251.6	15105983	33099.01	71686.12	1196049	2343498	33210.97
	best	32845.9	1676220	33319.29	259956	7015564	33025.92	32970.43	504395.5	975713.9	33056.59
	worst	33355.5	11010743	33357.15	366133.5	21307952	33236.13	181091.8	3081288	6129448	33341.95
	std	735.3243	13003315	53.22554	156632.1	18700060	306.6529	225673.2	3890551	7814528	363.3435
	median	32926.57	8983074	33325.33	357458.4	16050208	33066.99	36341.11	599257.2	1134415	33222.67
	rank	1	9	4	6	10	2	5	7	8	3
C11-F16	mean	142346.6	82346776	96261863	16236991	1.01E+08	144476.5	147870.4	1255853	2362861	149494.6
	best	139600.4	41690633	78051060	3329962	81711778	141362.8	138757.7	351982.8	551007.8	137102.2
	worst	148358.1	1.18E+08	1.06E+08	29355266	1.16E+08	146619	153125.3	3029758	5916825	157084
	std	12769.92	98859401	39475472	38169189	50425013	7047.958	19659.74	3723153	7459576	28498.12
	median	140713.9	85064328	1E+08	16131368	1.04E+08	144962.1	149799.2	820835.2	1491805	151896.2
	rank	1	8	9	7	10	2	3	5	6	4
C11-F17	mean	23732524	2.2E+10	1.91E+10	1.54E+10	1.88E+10	2571105	1.18E+10	1.24E+10	1.91E+10	1.71E+09
	best	2189146	2.16E+10	9.61E+09	1.48E+10	1.37E+10	2033318	9.63E+09	1.02E+10	1.37E+10	1.27E+09
	worst	86187869	2.29E+10	2.74E+10	1.59E+10	2.33E+10	3918833	1.5E+10	1.43E+10	2.33E+10	2.18E+09
	std	1.29E+08	1.86E+09	2.75E+10	1.53E+09	1.33E+10	2790936	7.78E+09	5.78E+09	1.27E+10	1.35E+09
	median	3276542	2.19E+10	1.96E+10	1.54E+10	1.9E+10	2166135	1.12E+10	1.25E+10	1.96E+10	1.69E+09
	rank	2	10	9	6	7	1	4	5	8	3
C11-F18	mean	1140586	1.27E+08	1.93E+08	12059376	29999545	1371225	6148190	74411318	1.45E+08	1964841
	best	944829	1.25E+08	1.62E+08	4337086	26692713	964148.1	2476787	51777147	1E+08	1433852
	worst	1710349	1.3E+08	2.25E+08	21148311	37722612	1997242	13654590	85224046	1.66E+08	2426315
	std	1174309	6071440	80983743	24246031	16022522	1521402	15741514	48193133	94907917	1386199
	median	953583.8	1.27E+08	1.93E+08	11376053	27791428	1261755	4230692	80322040	1.57E+08	1999599
	rank	1	8	10	5	6	2	4	7	9	3
C11-F19	mean	1457088	1.26E+08	1.87E+08	15386506	36627775	1713154	6656123	73223697	1.42E+08	2739856
	best	971070.6	1.23E+08	1.65E+08	6269686	35220614	1186669	3800094	63429417	1.23E+08	2528283
	worst	2392422	1.32E+08	2.14E+08	25190986	37751648	2190497	8514079	90924153	1.79E+08	3108545
	std	1969928	12261246	69396356	24625035	3240764	1298504	6596598	39369604	80694499	784167
	median	1232430	1.25E+08	1.85E+08	15042676	36769418	1737725	7155158	69270610	1.34E+08	2661298
	rank	1	8	10	5	6	2	4	7	9	3
C11-F20	mean	965302.5	1.29E+08	1.85E+08	14249005	36996435	1011367	12328429	79885010	1.54E+08	2131601
	best	945467.2	1.25E+08	1.64E+08	11634185	27832008	965605.7	5133946	68703680	1.34E+08	1729714
	worst	1021021	1.35E+08	1.96E+08	18719456	46698604	1118122	29295891	93388143	1.83E+08	2472265
	std	114865.7	14090507	43582020	10281044	23887837	222321.2	35213762	32005288	63585305	977457.4
	median	947361	1.28E+08	1.9E+08	13321189	36727564	980869.2	7441939	78724108	1.49E+08	2162212
	rank	1	8	10	5	6	2	4	7	9	3
C11-F21	mean	21.71428	118.8505	124.3066	37.33435	114.8553	25.57349	42.25092	64.83181	91.01355	36.24951
	best	18.03225	99.70601	102.9999	28.9105	91.51785	20.59068	35.38711	50.11577	66.45898	28.83694
	worst	27.375	141.9425	134.4526	45.48978	153.603	32.87336	54.89326	76.38872	115.651	43.21162
	std	12.32653	57.01754	44.432	24.19802	84.83981	16.44161	27.0804	34.03985	67.19941	18.18318
	median	20.72495	116.8767	129.887	37.46856	107.1502	24.41497	39.36165	66.41138	90.97211	36.47475
	rank	1	9	10	4	8	2	5	6	7	3
C11-F22	mean	23.94115	96.9244	89.70018	64.5387	122.637	24.46643	44.98745	56.1402	74.30445	33.30161
	best	21.84546	53.52053	68.53573	46.3531	115.1157	17.7902	40.36256	43.26338	52.03457	30.57862
	worst	26.0763	125.7335	108.3948	79.73511	129.8458	29.55522	50.15289	63.25966	85.38791	35.57484
	std	5.339952	98.64324	55.24547	51.67589	18.68147	15.85684	14.80194	27.28404	46.69579	7.51797
	median	23.92142	104.2218	90.93507	66.0333	122.7933	25.26015	44.71717	59.01888	79.89766	33.5265
	rank	1	9	8	6	10	2	4	5	7	3
Sum rank		26	167	151	117	160	63	101	142	184	91
Mean rank		1.181818	7.590909	6.863636	5.318182	7.272727	2.863636	4.590909	6.454545	8.363636	4.136364
Total rank		1	9	7	5	8	2	4	6	10	3
Wilcoxon: *p*-value			6.14E-17	7.18E-14	3.56E-15	2.21E-17	1.08E-05	8.07E-16	9.72E-18	2.14E-18	4.86E-15

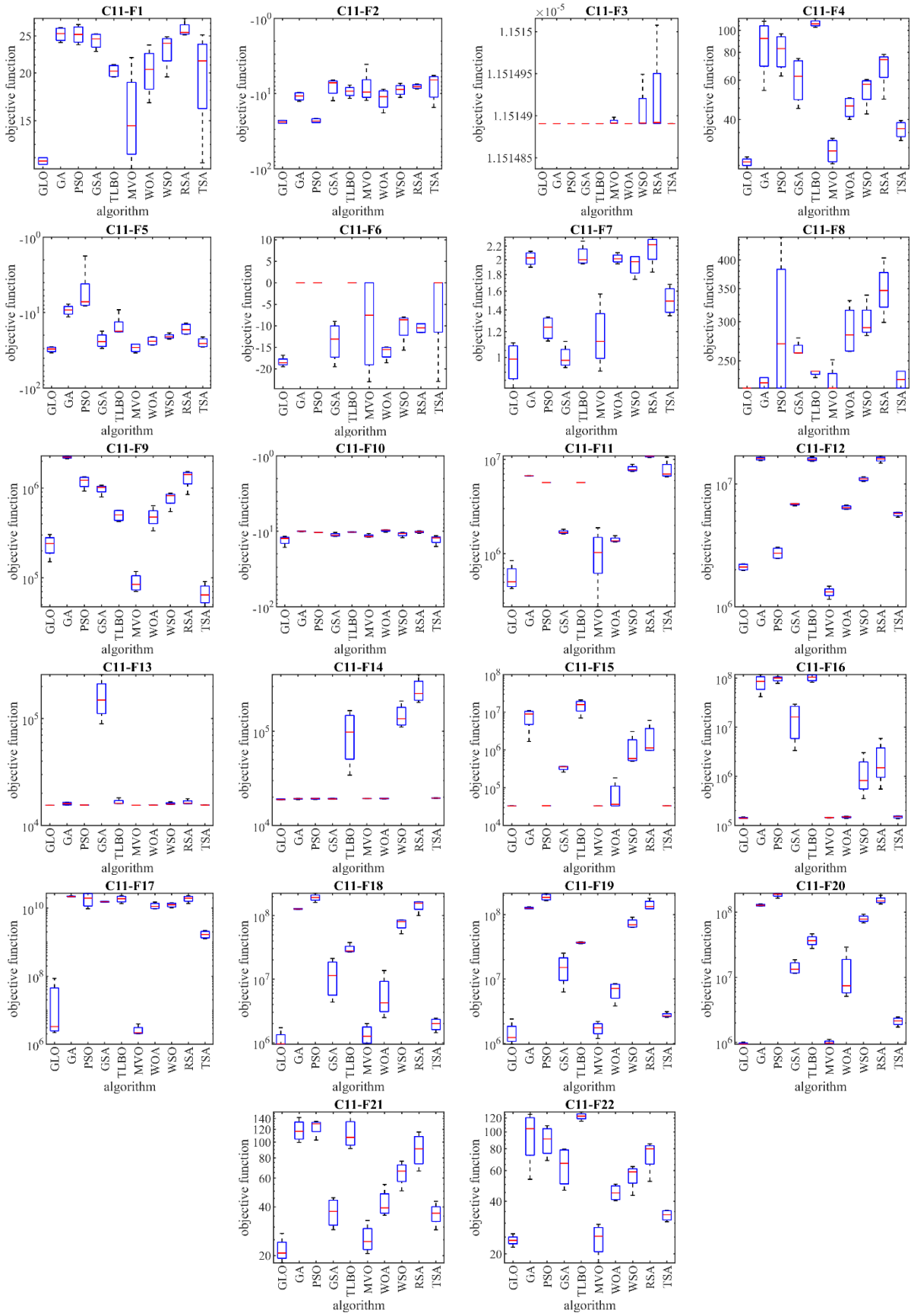

Figure. 1 Boxplot diagrams obtained from GLO and competing algorithms on CEC 2011 test suite

3.3 Discussion and interpretation of algorithm performance

A synthesis of the results indicates that the GLO algorithm achieved first rank in 19 out of 22 problems and, with a success rate of approximately 86.36%, is introduced as the best optimizer in the study. The results of the Wilcoxon test also confirm the statistically significant superiority of the algorithm in pairwise comparisons.

This successful performance can be directly attributed to the intrinsic design characteristics of the algorithm. First, the exploratory search pattern based on long-range creeping displacements and spiral trajectories enables extensive coverage of the search space and prevents premature convergence. Second, the exploitative mechanism based on short, oscillatory, and concentrated movements allows precise refinement of solutions around promising regions. The integration of these two behaviors establishes an effective balance between global and local search.

Moreover, the presence of oscillatory components, adaptive coefficients, and the gradual reduction of movement amplitude throughout the iterations causes the algorithm to operate exploratively in the early stages and progressively shift towards precise convergence. This gradual transition is one of the key factors in achieving both high accuracy and stability simultaneously.

Despite these advantages, the proposed algorithm is not without limitations. Similar to many population-based algorithms, its computational cost increases in very high-dimensional problems. Its performance may also be somewhat dependent on population size and the number of iterations. In problems with extremely strict constraints and very small feasible regions, additional tuning or auxiliary mechanisms may be required.

Nevertheless, the overall results demonstrate that, through intelligent management and balanced regulation of exploration and exploitation, GLO constitutes an effective and reliable approach for solving real-world constrained optimization problems. Its ability to produce accurate, stable, and repeatable solutions establishes its position as a competitive and efficient method among metaheuristic optimizers.

4. Conclusion and future research directions

In this paper, a new swarm-based metaheuristic algorithm named Glass Lizard Optimization (GLO) was introduced and developed, with its primary inspiration drawn from the behavioral patterns of the glass lizard reptile in nature. The conceptual foundation of the algorithm's design was established through the extraction and abstraction of two distinctive behaviors of this reptile: first, long-range creeping displacements accompanied by spiral trajectories during environmental exploration, and second, short, concentrated, and oscillatory movements during the final approach toward prey. Following their ethological interpretation, these two behavioral patterns were systematically formulated within two independent phases—exploration and exploitation—to provide a balanced mechanism for global and local search in the problem-solving space.

To evaluate the efficiency of the proposed algorithm, the performance of GLO in solving real-world application problems was investigated on 22 constrained optimization problems from the CEC 2011 test suite, and the obtained results were compared with nine competing algorithms. The optimization results demonstrated that, owing to its strong capability in establishing an effective balance between exploration and exploitation, GLO guides the search process in a purposeful manner and is able to attain high-quality solutions. This capability enabled the proposed algorithm to deliver superior performance over competitors in a substantial portion of the test problems.

A detailed examination of the comparative results revealed that GLO succeeded in achieving the first rank as the best optimizer in 19 out of the 22 problems, corresponding to a success rate of 86.36%. This level of performance reflects the behavioral stability and high efficiency of the algorithm when dealing with complex constrained problems. Furthermore, the results of the Wilcoxon statistical test confirmed the statistically significant superiority of the proposed algorithm over rival methods, indicating that the observed improvements were not merely due to randomness.

The findings of this study suggest that GLO can be employed as an efficient optimization tool for addressing a wide spectrum of engineering, scientific, and real-world application problems. Its conceptually simple structure, combined with coherent mathematical modeling, facilitates its implementation and extensibility across various domains.

Finally, several research directions are suggested for future developments. These include the design of binary and multi-objective versions of the algorithm, which could broaden its range of applications. In addition, applying GLO to very high-dimensional optimization problems, dynamic environments, and interdisciplinary applications across different scientific fields represents other valuable avenues for

future research. Such directions can further enhance the algorithm's performance and consolidate its position among modern optimization methods.

Conflicts of Interest

The authors declare no conflict of interest."

Author Contributions

Conceptualization, T.H, O.A.S, F.A.J, and A.S; methodology, M.D, Z.M, M.I, R.A.Z, and A.S; software, K.E, and R.A.Z; validation, K.E, F.A.J, and M.D; formal analysis, T.H, A.S, Z.M, and O.A.S; investigation, O.A.S, F.A.J, A.S, and T.H; resources, Z.M, M.I, R.A.Z, and T.H; data curation, Z.M and F.A.J; writing—original draft preparation, O.A.S, R.A.Z, and A.S; writing—review and editing, T.H, F.A.J, M.I, and K.E; visualization, K.E, R.A.Z, and M.I; supervision, M.D and Z.M; project administration, K.E, T.H, and O.A.S; funding acquisition, K.E.

References

[1] B. Maiti *et al.*, "Enhanced crayfish optimization algorithm with differential evolution's mutation and crossover strategies for global optimization and engineering applications", *Artificial Intelligence Review*, Vol. 58, No. 3, p. 69, 2025, doi: 10.1007/s10462-024-11069-7.

[2] D. A. Judeh and M. A. Hammad, "Applications of conformable fractional pareto probability distribution", *Int. J. Advance Soft Compu. Appl*, Vol. 14, No. 2, pp. 115-124, 2022.

[3] M. Elbes, T. Kanan, M. Alia, and M. Ziad, "COVD-19 Detection Platform from X-ray Images using Deep Learning", *International Journal of Advances in Soft Computing & Its Applications*, Vol. 14, No. 1, pp. 196-211, 2022, doi: 10.15849/IJASCA.220328.13.

[4] V. Basetti, C. K. Shiva, S. Tiwari, R. R. Karri, Y. Arya, and S. Sen, "Solving nonlinear engineering problems using UFIA: A social inspired metaheuristic algorithm", *Chaos, Solitons & Fractals*, Vol. 207, p. 117876, 2026, doi: 10.1016/j.chaos.2026.117876.

[5] K. Khlie, A. Pugalenthi, Z. Benmamoun, W. Aribowo, and M. Dehghani, "Sustainable Supply Chain Optimization: A Breakthrough in Swarm-based Artificial Intelligence", *Engineering, Technology & Applied Science Research*, Vol. 15, No. 3, pp. 23125-23132, 2025, doi: 10.48084/etasr.10505.

[6] W. F. Mbasso, A. Harrison, I. Dagal, P. Jangir, R. Kumar, and Z. Liu, "Fractional calculus-inspired metaheuristic algorithm for solving high-dimensional constrained optimization problems", *Evolutionary Intelligence*, Vol. 19, No. 1, p. 2, 2025, doi: 10.1007/s12065-025-01114-x.

[7] M. F. Şahin and F. Anka, "An adaptive hybrid metaheuristic algorithm for satellite images in remote sensing image segmentation", *The Visual Computer*, Vol. 42, No. 2, p. 134, 2026, doi: 10.1007/s00371-025-04341-6.

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

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization", In: *Proc. of ICNN'95 - International Conference on Neural Networks*, Vol. 4, pp. 1942-1948 1995, doi: 10.1109/ICNN.1995.488968.

[10] L. Wei and H. Zhong, "Falco peregrinus optimization algorithm: A novel biomimetic metaheuristic algorithm for engineering applications", *Energy Reports*, Vol. 15, p. 109064, 2026, doi: 10.1016/j.egyr.2026.109064.

[11] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization", *IEEE Computational Intelligence Magazine*, Vol. 1, No. 4, pp. 28-39, 2006, doi: 10.1109/MCI.2006.329691.

[12] E.-S. M. El-kenawy *et al.*, "Glider snake optimizer (GSO): a nature-inspired metaheuristic algorithm for global and engineering optimization problems", *Artificial Intelligence Review*, Vol. 59, No. 3, p. 91, 2026, doi: 10.1007/s10462-026-11504-x.

[13] H. Qawaqneh *et al.*, "Bolas Spider Algorithm: A Novel Efficient Nature-Inspired Metaheuristic for Complex Continuous Optimization", *International Journal of Intelligent Engineering & Systems*, Vol. 19, No. 1, pp. 221-239, 2026, doi: 10.22266/ijies2026.0131.14.

[14] D. E. Goldberg and J. H. Holland, "Genetic Algorithms and Machine Learning", *Machine Learning*, Vol. 3, No. 2, pp. 95-99, 1988/10/01 1988, doi: 10.1023/A:1022602019183.

[15] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341-359, 1997.

[16] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm", *Information Sciences*, Vol. 179, No. 13, pp. 2232-2248, 2009, doi: 10.1016/j.ins.2009.03.004.

[17] F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, and S. Mirjalili, "Henry gas solubility optimization: A novel physics-based algorithm", *Future Generation Computer Systems*, Vol. 101, pp. 646-667, 2019, doi: 10.1016/j.future.2019.07.015.

[18] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, Vol. 220, No. 4598, pp. 671-680, 1983.

[19] J. L. J. Pereira, M. B. Francisco, C. A. Diniz, G. Antônio Oliver, S. S. Cunha, and G. F. Gomes, "Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization", *Expert Systems with Applications*, Vol. 170, p. 114522, 2021, doi: 10.1016/j.eswa.2020.114522.

[20] I. Matoušová, P. Trojovský, M. Dehghani, E. Trojovská, and J. Kostra, "Mother optimization algorithm: a new human-based metaheuristic approach for solving engineering optimization", *Scientific Reports*, Vol. 13, No. 1, p. 10312, 2023, doi: 10.1038/s41598-023-37537-8.

[21] J. Rajabov *et al.*, "Dance Training-based Optimizer: A Novel Human-inspired Metaheuristic Algorithm for Optimization Tasks", *International Journal of Intelligent Engineering & Systems*, Vol. 19, No. 2, pp. 754-764, 2026, doi: 10.22266/ijies2026.0228.47.

[22] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems", *Computer-Aided Design*, Vol. 43, No. 3, pp. 303-315, 2011, doi: 10.1016/j.cad.2010.12.015.

[23] A. Zraiqat *et al.*, "Key Maker Algorithm: A Novel Human-Based Metaheuristic for Constrained Optimization", *International Journal of Intelligent Engineering & Systems*, Vol. 18, No. 9, pp. 688-699, 2025, doi: 10.22266/ijies2025.1031.45.

[24] A. Zraiqat *et al.*, "Detective Algorithm: A Novel Human-inspired Metaheuristic for Solving Real-world Constrained Optimization Problems", *International Journal of Intelligent Engineering & Systems*, Vol. 18, No. 10, pp. 33-46, 2025, doi: 10.22266/ijies2025.1130.03.

[25] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems", *Jadavpur University, Nanyang Technological University, Kolkata*, pp. 341-359, 2010.

[26] S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization", *Engineering Applications of Artificial Intelligence*, Vol. 90, p. 103541, 2020, doi: 10.1016/j.engappai.2020.103541.

[27] L. Abualigah, M. A. Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer", *Expert Systems with Applications*, Vol. 191, p. 116158, 2022, doi: 10.1016/j.eswa.2021.116158.

[28] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm", *Advances in Engineering Software*, Vol. 95, pp. 51-67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[29] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-Verse Optimizer: a nature-inspired algorithm for global optimization", *Neural Computing and Applications*, Vol. 27, No. 2, pp. 495-513, 2016, doi: 10.1007/s00521-015-1870-7.

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