Analytical and Numerical Solutions for Nonlinear Equations
©Available online at https://ansne.du.ac.ir/
Online ISSN: 3060–785X
2026, Volume 11, Issue 1, pp. 51–83
Research Article

Pirates of the Caribbean Metaheuristic: A Novel Optimization Algorithm Inspired by Cinematic Metaphors for Solving Complex Optimization Problems

Yasin Alipour[1,*], Abdolali Basiri[1], Reza Pourgholi[1], Mahmoud Moallem[1], Hossein Mollajafary[1]

[1] School of Mathematics and Computer Science, Damghan University, Damghan, Iran

* **Corresponding author(s):** ya.alipour@du.ac.ir

Received: 19/01/2026 **Revised:** 10/03/2026 **Accepted:** 17/03/2026 **Published:** 07/06/2026 10.22128/ansne.2026.3221.1187

Abstract

This study introduces a novel population-based metaheuristic algorithm, the Pirates of the Caribbean Optimization Algorithm (PCOA), inspired by the adventurous strategies of pirates in cinematic narratives. PCOA models the exploration and exploitation process through the metaphor of pirate crews searching for hidden treasure while navigating unpredictable challenges. The algorithms effectiveness is evaluated by extensive comparisons with leading optimization methods across 23 classical functions and the CEC 2019 benchmark suites. Results consistently demonstrate PCOAs superior solution quality, robustness, and convergence speed. Additionally, PCOA is successfully applied to challenging real-world inverse problems in nonlinear partial differential equations, highlighting its practical potential. The open-source implementation of PCOA further supports reproducibility and future research.

Keywords: Pirates of the Caribbean optimization algorithm (PCOA), Cinematic Metaphor, Hybrid metaheuristic, Global optimization, Partial differential equations (PDEs)

Mathematics Subject Classification (2020): 65Mxx, 90C26

1 Introduction

solutions where classical methods such as Hill Climbing [1], Simulated Annealing [2], and Nelder-Mead [3] frequently struggle to converge efficiently or become trapped in local minima [4]. The appeal of metaheuristics lies in their ability to explore complex and multimodal search spaces effectively, making them invaluable for addressing computationally challenging and real-world optimization problems [5]. Motivated by these advantages, researchers have introduced a wide array of metaheuristic frameworks inspired by natural, physical, and cultural phenomena, broadening the horizons of optimization methodologies [6]. Despite remarkable progress, many existing metaheuristic algorithms still encounter critical challenges, including premature convergence, insufficient exploration of the search space, or limited performance when tackling highly multimodal or dynamic optimization landscapes. These issues highlight the ongoing need for the development of more effective and adaptable optimization strategies.

In this work, we propose a novel population-based metaheuristic algorithm, termed the Pirates of the Caribbean Optimization Algorithm

(PCOA). The PCOA draws inspiration from the adventurous pursuits of pirate ships seeking hidden treasures, modeling the optimization process as multiple ships, each manned by a crew of pirates, navigating a vast ocean (the solution space) and searching various islands (candidate solutions) in quest of the elusive treasure island representing the global optimum (see Fig. 1). To enhance its optimization capability, PCOA incorporates several advanced mechanisms, including chaotic reflection, Lévy flights, quasi-opposition learning, elitism, differential evolution (DE)-style crossover, and local refinement. These strategies collectively enable the algorithm to achieve a better balance between exploration and exploitation, mitigating the risk of premature convergence and ensuring a comprehensive search of the solution landscape. Established metaheuristic methodologies such as Genetic Algorithms (GA) [7], Ant Colony Optimization (ACO) [8], and Particle Swarm Optimization (PSO) [9] have demonstrated notable success across diverse scientific and engineering domains. The sustained prominence of these approaches is largely due to their algorithmic simplicity, structural adaptability, and derivative-free search capabilities, which enable them to handle intricate and black-box optimization problems effectively [10, 11]. Metaheuristics are inherently intuitive and conceptually accessible [5, 12, 13], and their inspirations from nature or evolution [14] facilitate their rapid adoption and continuous theoretical development across disciplines [12, 15]. Their high adaptability allows them to be applied to a broad spectrum of optimization problems with minimal modifications [16], especially as they emphasize input-output relationships over system internals [17], [18]. Furthermore, the stochastic search strategies and derivative-free nature of metaheuristics [10,19] make them particularly well-suited for real-world problems where derivatives are difficult or expensive to obtain. Crucially, their inherent stochasticity enables them to circumvent local optima and explore complex, multimodal landscapes more effectively than traditional optimization methods [10].

It should be noted that, according to the No Free Lunch (NFL) theorem, no universal metaheuristic can consistently outperform all others across every problem domain [20]. This fundamental principle motivates continuous innovation and refinement in metaheuristic research to address the diverse and evolving challenges encountered in real-world optimization tasks. Metaheuristic algorithms are generally classified into two categories: solution-based (trajectory-based) methods and population-based methods. While the former, such as simulated annealing, operates with a single candidate solution [20], the latter leverages multiple solutions, promoting information exchange and collective search to better explore the solution space [20,21]. Population-based metaheuristics, including swarm intelligence algorithms like ACO, PSO, and Artificial Bee Colony (ABC) [22–24], have demonstrated remarkable robustness and flexibility across a range of complex applications, from network design [25] to combinatorial optimization [26]. Their success is attributed to their collective search dynamics, adaptability, and the effective balance they maintain between exploration and exploitation [24, 27].

Despite the diversity among metaheuristic approaches, the search process is typically structured into two fundamental phases: exploration and exploitation [12, 28]. Achieving an effective balance between these phases remains a central challenge, as excessive exploration can slow convergence while excessive exploitation can lead to premature stagnation [29, 30]. Furthermore, when applied to inverse boundary identification for the Fisher and Huxley equations, inspired by real-world engineering scenarios such as inverse heat conduction problems (IHCPs) [10,31], PCOA exhibits high convergence speed and accuracy in reconstructing unknown parameters, further demonstrating its effectiveness for challenging scientific and engineering tasks.

2 Strategic Behaviors in Cinematic Narratives and Optimization

Cinematic storytelling, particularly within high-stakes adventure narratives, introduces a sophisticated blend of exploration, strategic maneuvering, and adaptive learning, mirroring the challenges faced in global optimization. Unlike purely stochastic or biologically emergent search processes, cinematic narratives incorporate intentionality, overarching strategic arcs, and multi-layered decision-making, factors that can significantly enhance heuristic search methodologies. Several key components of this paradigm include:

1. **Dynamic Exploration and Exploitation:** In pirate-themed narratives, ships embark on perilous voyages across unpredictable seas in search of hidden treasure. This mirrors the fundamental optimization challenge of navigating complex, multimodal search landscapes, balancing broad exploration to avoid local optima while refining search efforts near promising solutions. In the PCOA, this cinematic metaphor is directly implemented through two key algorithmic phases. During the early exploration stage, each agent (pirate ship) samples widely across the solution space by updating its position based on random vectors and the trajectories of other ships (see Equation (5)). As the search progresses, the algorithm transitions into a more exploitative mode, refining solutions by focusing updates around the best-known locations and gradually reducing the degree of randomness (see Equation (9)). These mathematically defined phases ensure that the algorithm initially explores globally to discover promising regions and then exploits locally to converge towards the global optimum, mirroring the pirates need to balance searching new waters with investigating the most promising islands.

2. **Collaborative Intelligence and Strategic Communication:** Pirate fleets frequently exchange intelligence, form temporary alliances, and employ deceptive strategies, analogous to swarm intelligence mechanisms and differential evolution strategies that enhance search efficiency through collective adaptation (section 2.3.6.4).

3. **Environmental Perturbations and Adaptive Navigation:** The unpredictability of the sea, ranging from violent storms to treacherous reefs, forces navigators to dynamically recalibrate their trajectories in real-time. This concept is directly translated into controlled perturbations within the optimization algorithm, designed to prevent premature convergence and ensure robust adaptability. In the Pirates of the Caribbean Optimization Algorithm (PCOA), this narrative translates to the incorporation of random perturbation and mutation mechanisms, such as sinusoidal perturbations and Gaussian quasi-reflection, at various stages of the search process. These mathematical operations, detailed in Equations (18) and (section 2.3.6), introduce stochastic variations into the positions of the agents (pirate ships), helping the population escape local optima and enhancing overall adaptability. By periodically applying these controlled disturbances, the algorithm prevents premature convergence and ensures that the exploration of the solution space remains diverse and resilient, much like a fleet of ships dynamically steering through unpredictable and stormy seas.

4. **Hybrid Search Maneuvers:** Adventure narratives frequently depict bold, sweeping movements (akin to flights) interwoven with precise, localized refinements (resembling gradient-based searches), striking a nuanced balance between global exploration and local exploitation, a crucial factor in high-performance metaheuristic design.
   Within the algorithm, this metaphor is implemented by alternating between global moves using Lévy flights and local refinement strategies near the best solutions. Equation (11) and section 2.3.8 formalize these search maneuvers, allowing agents to make both large exploratory jumps and fine-tuned local adjustments, thus balancing diversification and intensification throughout the optimization process.

2.1 Cinematic Metaphors as a Paradigm for Metaheuristic Optimization

Metaheuristic optimization algorithms have become indispensable for tackling complex, high-dimensional, and multimodal optimization problems. By leveraging stochastic principles derived from biological evolution, swarm intelligence, and physical processes, these algorithms efficiently explore vast search spaces to identify optimal solutions. However, despite the significant advancements brought by nature-inspired methodologies, they often remain constrained by their reliance on emergent, localized interactions and stochastic perturbations. This limitation raises a critical question: can structured, human-driven decision-making frameworks enhance metaheuristic search dynamics?

2.2 Algorithmic Design and Key Innovations

By systematically translating these cinematic principles into a computational framework, the proposed algorithm introduces multiple innovative mechanisms that enhance search efficiency and robustness:

- Elitism and Solution Retention: Inspired by the strategic decision-making processes of successful pirate captains, the algorithm employs an elitism-based selection protocol that preserves superior solutions across generations, ensuring progressive improvement while mitigating performance degradation.

- Multi-Agent Communication for Enhanced Exploration: Just as pirate fleets coordinate intelligence gathering and divide search responsibilities, the algorithm enables search agents to exchange information dynamically, fostering a collaborative learning environment that accelerates convergence and enriches solution diversity.

- In PCOA, the simulation of sudden oceanic storms serves as a dedicated strategy to introduce adaptive perturbations, enabling search agents to effectively escape local optima and enhance the algorithms ability to explore alternative regions of the search space.

2.3 Mathematical Modeling

2.3.1 Algorithm Preliminary Process

The proposed Pirates of the Caribbean Optimization Algorithm (PCOA) is a population-based algorithm, where pirate ships serve as the exploratory agents within the framework. In this algorithm, each population member represents a potential solution to the problem and

Figure 1. A cinematic-inspired metaheuristic approach is illustrated, drawing thematic influence from the film Pirates of the Caribbean. Image source: Promotional material from Amazon.com.

defines the values of the variables. In PCOA, the position of each pirate ship in the search space directly represents the values of the decision variables. These positions serve as candidate solutions to the problem being solved.

During the initial stage of PCOA, Equation (1) is applied to randomly initialize the positions of the pirate ships within the specified search boundaries:

$$X_{i,j}^{(1)} = lb_j + \text{rand}() \times (ub_j - lb_j), \qquad i = 1,2,3,\ldots,N, \quad j = 1,2,3,\ldots,\text{dim} \tag{1}$$

here, X_i is the position of the $i-th$ pirate ship, lb_j and ub_j denote the lower and upper bounds for the $j-th$ dimension, respectively, and $rand(\,)$ is a random function uniformly generate distributed numbers between 0 and 1.

This formulation ensures that each dimension of every ship's position is sampled uniformly within the allowable bounds, maintaining diversity and preventing initial bias in the population. This diversity lays the foundation for robust exploration of the search space during subsequent algorithmic phases.

Mathematically, each population member is modeled as a vector, and collectively, these vectors constitute the population matrix of the algorithm. Initially, the population members are randomly distributed within the search space. The population matrix in PCOA is generated as described by Equation (2):

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,\text{dim}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,\text{dim}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,\text{dim}} \end{pmatrix}_{N\times\text{dim}}, \tag{2}$$

where, X is the population matrix representing the positions of all pirate ships; X_i is the i-th proposed solution (position of the i-th pirate ship); $x_{i,j}$ represents the value of the j-th variable defined by the i-th proposed solution; N is the total number of population members (pirate ships) and dim is the number of problem variables (dimensions).

Each population member represents a proposed solution to the problem. Accordingly, the objective function of the problem can be

evaluated for each population member. The resulting values of the objective function can be expressed as a vector using Equation (3):

$$F(X) = \begin{pmatrix} f_1(X_1) \\ \vdots \\ f_i(X_i) \\ \vdots \\ f_N(X_N) \end{pmatrix}_{N\times 1}, \qquad f_i(X_i) = \text{fitness}(X_i), \tag{3}$$

here, F is a vector containing all the objective function values obtained from the population members; and f_i represents an objective function value calculated for the i-th proposed solution within the population.

We denote the best solution (the best-known treasure location) and its fitness as:

$$B^* = \arg\min_i \{X_i\}, \qquad f^* = \min_i \{f_i\}, \tag{4}$$

2.3.2 Search Phases

The operation of the Pirates of the Caribbean Optimization Algorithm (PCOA) is structured into three primary search phases, each corresponding to a specific stage of the treasure hunting process. These phases are designed to ensure a balanced trade-off between exploration and exploitation, gradually refining the search for the optimal solution as iterations progress. The cinematic metaphor of pirate ships navigating an oceanic search space in pursuit of a treasure island serves as the foundation for these phases, aligning with the narrative structure of "Pirates of the Caribbean." Here, pirate ships navigate through challenges such as turbulent waters, hidden reefs, and competition among themselves, reflecting the strategic progression of the treasure hunt. The three-phase structure reflects the strategic progression of the treasure hunt:

- **Exploration Phase:** Ships embark on perilous voyages across unpredictable seas, symbolizing the need for extensive exploration in complex optimization landscapes.
- **Approaching Phase:** As ships identify promising regions, they refine their search efforts while maintaining adaptability, mirroring the balance between exploration and exploitation.
- **Convergence Phase:** In the final stages, ships employ precise navigation techniques to locate the treasure island, representing the algorithm's ability to converge toward the global optimum.

2.3.3 Initial Search Stage (Exploration Phase)

In the early exploration phase ($t < T/3$), each pirate ship adjusts its course by combining its own direction with information gleaned from the trajectories of other ships, guided by unpredictable winds. This collective navigation strategy enhances the fleets ability to uncover promising regions within the search space and is mathematically expressed as:

$$X_i^{(t+1)} = X_i^{(t)} + \left(X_r^{(t)} - X_s^{(t)}\right) \odot R_1, \tag{5}$$

here, $X_i^{(t+1)}$ is the updated position of ship i, $X_r^{(t)}$, $X_s^{(t)}$ are two randomly selected ships from the population, $R_1 \in \mathbb{R}^d$ is a random vector with elements uniformly distributed in $[0,1]$ and $\odot$ represents element-wise multiplication. This update mechanism leverages the diversity between randomly selected ships, enabling the algorithm to explore uncharted regions of the search space and prevent premature convergence.

2.3.4 Approaching Treasure Stage (Approaching Phase)

In the middle third of the iterations ($T/3 \leq t < 2T/3$), as some knowledge about promising regions is gained, each ship gradually aligns its course with the best-known location B^*. Brownian motion R_B is introduced at this stage to represent the ships stochastic navigation. We define a damping factor q_w to reduce the magnitude of random displacements. Additionally, this strategy not only mitigates the risk of premature convergence to local optima, but also enhances the algorithms convergence rate toward optimal solutions. By enabling individuals

to utilize both global information and their own historical best positions, the probability of identifying the global optimum increases. The incorporation of Brownian motion introduces stochasticity, further improving the algorithms ability to escape local traps and thoroughly explore the solution space. As a result, updating the ships position during the Approaching Treasure stage can be mathematically formulated as shown in Equation (7).

$$q_w = e^{-5p^2}, \qquad p = \frac{t}{T}, \tag{6}$$

$$X_{\text{new}}^{(t)} = B^* + q_w \times (R_B - 0.5) \times \left(B^* - X_i^{(t-1)}\right), \tag{7}$$

where, R_B is a random vector with elements uniformly distributed in $[\text{Low}_{\text{local}}, \text{High}_{\text{local}}]$, $\text{Low}_{\text{local}} = \frac{lb}{t+1}$ and $\text{High}_{\text{local}} = \frac{ub}{t+1}$ define dynamically adjusted bounds for local exploration. This phase enhances exploitation by refining the search around promising regions while maintaining some exploratory capacity through the randomness introduced by R_B. So, this equation ensures that ships move closer to the best-known solution B^*, while still allowing for some degree of randomness, preventing premature convergence. By multiplying the stochastic adjustment term $(R_B - 0.5) \times (B^* - X_i^{(t-1)})$ by the decaying weight q_w you ensure that both the direction and magnitude of each random step shrink over time according to the same schedule. Early on, when $q_w \approx 1$, the swarm makes larger, more exploratory moves around B^*; as t increases and q_w falls, these moves become progressively more conservative, honing in on the best-known solution. Because q_w follows a smooth, bell-shaped decay, the transition from exploration to exploitation is gradual rather than abrupt, which helps stabilize convergence and reduces oscillations around B^*. In practice, this multiplicative coupling of randomness and time-varying weight elegantly balances global search with local refinement, maintaining enough diversity to escape shallow traps early on, while concentrating computational effort on fine-tuning the optimum in later stages. Moreover, this mechanism is specifically aimed at identifying the best anchoring location, ensuring ships can find the safest and most optimal point at each phase of the algorithms progression.

2.3.5 Moving Towards Treasure Stage (Convergence Phase)

In the final third of the iterations ($t > 2T/3$), the proposed PCOA algorithm treats each parent solution as an anchor point for a localized, stochastic search inspired by Lévy flights. This behavior is modeled as:

$$CF = 1 - (t/T)^3 \times (10 - 15t/T + 6(t/T)^2), \tag{8}$$

$$X_{\text{new}}^{(t)} = B^* + CF \times X_i^{(t)} \odot L, \tag{9}$$

$$L = \text{LévyFlight}(\text{dim}, \beta), \tag{10}$$

here, B^* represents the best-known solution, L is a Lévy step vector, and CF is a convergence factor that diminishes step size as iterations progress. L combines small and large jumps for diverse search behavior.

$$L = \frac{u}{|v|^{1/\beta}}, \tag{11}$$

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, 1), \tag{12}$$

where β is a parameter controlling the heaviness of the distributions tail, adaptively set within the range $1 < \beta \leq 2$. The standard deviation σ_u is determined by:

$$\sigma_u = \left(\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma((1+\beta)/2)\beta 2^{(\beta-1)/2}} \right)^{1/\beta}, \tag{13}$$

where $\Gamma(\cdot)$ denotes the Gamma function. Instead of a constant β, the PCOA algorithm smoothly transitions β from 1.5 to 2.0 as t increases:

$$\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \times \left(1 - e^{-\sigma(t/T)^p}\right), \tag{14}$$

where $\beta_{\min} = 1.5$, $\beta_{\max} = 2.0$, and σ, p are parameters controlling the steepness and acceleration of the transition (e.g., $\sigma = 4.1$, $p = 1.5$) respectively. Once L is generated, it is scaled to a desired radius or used directly in:

Lévy flight is utilized to simulate the exploratory movement of pirate crew. Each pirate crew (the set of individuals on a ship) is capable of performing local island exploration via random jumps governed by Lévy flights. For a step size $L \in \mathbb{R}^d$. When a ship finds a better island (i.e., the position X_i improves its fitness), it dispatches its pirates as offspring around X_i. Let r be a small radius:

$$r = 0.1 \times (ub - lb) \times (1 - t/T), \tag{15}$$

$$\delta_m = \frac{L}{\|L\|} \cdot r, \tag{16}$$

$$X_{(i,\text{child})}^{(m)} = X_i^{\text{Parent}} + \delta_m, \quad m = 1, 2, \ldots, M, \tag{17}$$

where δ_m is a directional step is computed based on a reference vector. This formulation ensures controlled exploration along relevant directions while preserving stability in the absence of directional cues, a strategy commonly employed in nature-inspired metaheuristics.

2.3.6 Advanced Mechanisms in Every Phase

The Pirates of the Caribbean Optimization Algorithm (PCOA) incorporates several advanced mechanisms to enhance exploration, exploitation, and convergence. Below is a detailed explanation of these mechanisms, including their mathematical formulations and roles in the algorithm.

2.3.6.1 Sinusoidal Perturbation Mechanism

To prevent premature convergence and encourage global exploration, a sinusoidal perturbation mechanism is applied. This introduces periodic oscillations into the search process, ensuring that ships do not get trapped in local optima. The perturbation value is calculated as:

$$P_i^{(t)} = \left(\alpha_{\text{initial}} e^{\beta_P t/T} + \alpha_{\text{mid}} e^{-\gamma(t/T - 0.5)^2} \right) \cdot (1 + \delta \sin(2\pi t/T)) \cdot pf, \tag{18}$$

where α_{initial} and α_{mid} are initial and mid-term perturbation factors, β_P, γ, δ are parameters controlling the perturbation dynamics, and $pf = 1/(1 + \eta(t/T)^p)$ is the probability factor. The sinusoidal component $(2\pi t/T)$ introduces oscillatory behavior, while the exponential decay terms ensure that perturbations diminish over time, promoting convergence.

$$X_{\text{new}}^{(t)} = \begin{cases} \begin{cases} 1 + \delta \sin(2\pi t/T), & \text{if } r_2 < 0.5, \\ B^* + P_i^{(t)} \times (2 \times R_B - 1) \times X_i, & \text{if } f^* \text{ is not satisfied}, \end{cases} & r_1 < 0.5, \\ \begin{cases} e^{-5t/T}, & \\ |B^{*2} - X_i^2| \times e^{5 \cdot \text{rand}(-1,1)} \cdot \cos(2\pi \cdot \text{rand}(-1,1)), & \text{if } f^* \text{ is not satisfied}, \end{cases} & r_1 \geq 0.5, \end{cases} \tag{19}$$

where r_1 is a random number generated at each step to decide the main update condition and r_2 is a secondary random number for finer decision-making within the first branch.

2.3.6.2 Gaussian Quasi-Reflection

To expand the search space further, Gaussian quasi-reflection is applied. This technique creates quasi-opposite and reflected solutions, ensuring diversity and preventing premature convergence. The primary goal of Gaussian quasi-reflection is to:

- Expand the search space by creating alternative solutions that are quasi-opposite or reflected with respect to the current position of the agent.
- Introduce diversity into the population, reducing the risk of stagnation or entrapment in local optima.
- Ensure robust exploration while maintaining adaptability as the algorithm progresses.

This mechanism is particularly effective during the early and intermediate phases of the optimization process when extensive exploration is critical.

To encourage further diversification, the algorithm employs a reflection mechanism whenever a new solution fails to improve its fitness or periodically as an additional move. For each ships position , we define:

1. Opposite Position (x_i^{op}): For each agent X_i, a quasi-opposite solution x_i^{op} is generated using the following formula:

$$x_i^{op} = lb + ub - X_i \times e^{\left(-\frac{t}{T}\right)(-1)^{t+1}}, \tag{20}$$

This is a time-dependent opposite point, factoring in an exponentially decaying term $e^{\left(-\frac{t}{T}\right)}$. Where, x represents the current position of an agent, t is the current iteration, and T is the total number of iterations. This equation introduces a time-dependent exponential factor $(e^{-1/T})$ and an alternating sign $((-1)^{(t+1)})$ to dynamically adjust the quasi-opposite solution based on the progress of the algorithm.

2. Quasi-Opposite Position (x_i^{qo}): A quasi-opposite approach restricts the new candidate between the midpoint $m = \frac{1}{2}(lb + ub)$ and the opposite point x_i^{op}:

$$x_i^{qo} = L_q + U_q\, rand(0,1), \tag{21}$$

$$L_q = \min(m, x_i^{op}), \quad U_q = \max(m, x_i^{op}) - L_q, \tag{22}$$

3. Gaussian Factor (γ): A $(0,1)$ sample is drawn from a standard normal distribution via the logistic function:

$$z \sim N(0,1), \quad \gamma = \frac{1}{1+e^{-z}}, \tag{23}$$

4. Quasi-Reflection (x_i^{gqo}): We use γ to shift X_i and form another reflected candidate:

$$r_i = lb + ub - \gamma X_i, \tag{24}$$

$$x_i^{gqo} = L_r + U_r\, rand(0,1), \quad L_r = \min(m, r_i), \quad U_r = \max(m, r_i) - L_r, \tag{25}$$

Each ship tests x_i^{qo} and x_i^{gqo} for improved fitness. If any yields a better fitness than X_i, an update is performed:

$$X_i \leftarrow \arg\min\{fitness(x_i^{qo}), fitness(x_i^{gqo})\}, \tag{26}$$

$$X_i \leftarrow \min\{fitness(x_i^{qo}), fitness(x_i^{gqo})\}, \tag{27}$$

If $fitness(X_i)$ improves upon the global best f^*, then $B^* = X_i$ and $f^* = fitness(X_i)$.

Gaussian quasi-reflection is integrated into the main loop of PCOA and is applied to all agents at each iteration. This ensures that the population remains diverse and adaptive throughout the optimization process. Specifically:

- During the exploration phase, it expands the search space by introducing quasi-opposite and reflected solutions.
- During the approaching phase, it refines the search by focusing on promising regions while maintaining diversity.
- During the convergence phase, it assists in fine-tuning the solution around the global optimum.

By incorporating Gaussian quasi-reflection, PCOA achieves a balanced trade-off between exploration and exploitation, making it well-suited for solving complex, multimodal optimization problems.

2.3.6.3 Crossover: Information Exchange Among Ships

The algorithm incorporates a Differential Evolution (DE)-style crossover mechanism to facilitate efficient information exchange among pirate ships, analogous to the sharing of intelligence about potential treasure locations. This mechanism enhances the diversity of solutions while directing the search towards promising areas. The crossover process involves a mutant vector, which is generated before combining with the target vector to create a trial vector. A mutant vector is generated as:

$$v = X_a + F \times (X_b - X_c), \tag{28}$$

where, X_a, X_b, X_c are randomly selected ships from the population (distinct from X_i), and F is the differential weight ($F = 0.5$ is the chosen differential weight) that controls the amplification of the difference vector.

The trial vector u is created by mixing the mutant vector with the target vector (X_i):

$$u_j = \begin{cases} v_j, & \text{if } rand() < C_r \text{ or } j = j_{\text{rand}}, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{29}$$

where, C_r is the crossover probability (e.g., 0.1), probability of replacing the target vector's component with the mutant vector's component. $rand()$ is a random number uniformly distributed between 0 and 1. j is dimension index and j_{rand} is a randomly chosen dimension index to ensure at least one component is taken from the mutant vector.

Finally, accept if $fitness(u) < fitness(x_i)$; otherwise discard.

2.3.7 Elitism Protocol: Elitism with Rollback

The algorithm incorporates an advanced Elitism with Rollback mechanism, meticulously engineered to ensure the preservation of superior-quality solutions while embedding a sophisticated flexibility framework to accommodate recovery from suboptimal updates. This mechanism uses a metaphor, wherein a fleet of pirate ships (symbolizing candidate solutions) engages in a nuanced interplay of competitive strategies and episodic cooperative dynamics, collectively navigating toward the coveted treasure, emblematic of the global optimum.

This dual-faceted mechanism emphasizes the strategic retention of elite solutions, analogous to safeguarding the navigational prowess of the most successful vessels charting optimal routes. Concurrently, it empowers the population to exhibit adaptive resilience by recalibrating and diversifying its exploratory trajectory in response to the stagnation of progress or the emergence of uncharted complexities within the search landscape.

By seamlessly integrating the exploitation of high-caliber solutions with the adaptive recalibration afforded by the rollback strategy, the mechanism achieves a sophisticated equilibrium, balancing convergence toward optimality with the maintenance of population diversity, thereby enhancing robustness and adaptability in dynamic optimization scenarios. This process is applied through the following steps:

To preserve valuable discoveries, an elitism operator is invoked, where a fraction μ (e.g., 10%) of the best ships (lowest fitness) are identified and used to replace the μ worst ships. Denote the set of indices of the μN best ships as $\mathscr{E}_{\text{best}}$ and then μN worst as $\mathscr{E}_{\text{worst}}$. Then:

$$x_k \leftarrow x_j, \quad k \in \mathscr{E}_{\text{worst}},\ j \in \mathscr{E}_{\text{best}}, \tag{30}$$

where each worst individual is replaced by a corresponding elite solution. However, if this replacement does not yield an improvement in the global best f^*, the entire update is rolled back to ensure no detrimental effect on the population diversity. Symbolically, if post-elitism $\min f_i \geq f^*_{\text{old}}$, revert:

$$x_i \leftarrow x_i^{\text{backup}}, \quad \forall i \in \{1, 2, \ldots, N\} \tag{31}$$

This ensures we do not lose beneficial diversity or the global best solution.

2.3.8 Local Refinement

After every few iterations, the best ship B^* undergoes a localized search to refine its position, akin to pirates scouring every corner of the discovered best island. Specifically, we select a neighborhood radius:

$$r_{\text{local}}(t) = (ub_j - lb_j) \times 0.1 \times \left(1 - \frac{t}{T}\right) \tag{32}$$

and generate a small number v of random directions δ_k around B^*. The best improvement, if found, updates B^*. This can be formulated by:

$$\delta_k = \theta_k \frac{g_k}{\|g_k\| + 10^{-30}}, \quad \delta_k \sim \mathscr{N}(0, I_d), \quad 0 \leq \theta_k \leq r_{\text{local}}(t), \tag{33}$$

Hence, the new candidate is:

$$B^*_{\text{new}} = B^* + \delta_k \tag{34}$$

accepting B^*_{new} if its fitness is an improvement.

2.4 Putting It All Together: The ‘‘Pirates of the Caribbean Optimization Algorithm’’

The complete workflow of one iteration (t) is summarized below and the flowchart is depicted in Figure 2:

1. **Evaluate Current Fitness:** For all ships x_i compute $f_i = \text{fitness}(x_i)$. Update global best B^* and f^*.
2. **Movement by Phase:**
   - If $t < T/3$, use random vector-based exploration (Stage 1).
   - If $T/3 \leq t < 2T/3$, apply approaching formula (Stage 2).
   - Otherwise, use Lévy-based exploitation around B^* (Stage 3).
3. **Check and Apply Quasi-Reflection:** Generate $x_i^{op}, x_i^{qo}, x_i^{gqo}$ and update x_i if they yield better fitness.
4. **Elitism with Rollback:** Replace worst μN solutions with best μN solutions. Keep the new population only if the global best improves or remains the same.
5. **DE-Style Crossover:** For each x_i, create a mutant vector from other random ships, perform binomial crossover, and update x_i if the trial vector is better.
6. **Local Refinement:** Periodically refine B^* by sampling neighbors and picking the best solution found.
7. **Update Global Best:** If x_i improves upon f^*, then $B^* = x_i$ and $f^* = \text{fitness}(x_i)$.
8. Repeat for $t = t + 1$ until $t > T$.

2.5 Computational Complexity Analysis

predominantly governed by the operations executed at each iteration. In every iteration, the evaluation of the objective function across all individuals in the population incurs a computational cost of $O(N)$ where N represents the number of pirate ships. Simultaneously, the update of each ships position within a D-dimensional search space necessitates $O(ND)$ operations, with D denoting the number of decision variables. The application of diversity-preserving mechanisms, including Gaussian quasi-reflection, differential evolution-style crossover, and stochastic mutation, further contributes to the computational burden, each operating at an order of $O(ND)$.

The elitism mechanism, which preserves superior solutions through selective replacement, involves either partial selection or full sorting operations, leading to a worst-case complexity of $O(N \log N)$. Nevertheless, given that D typically exceeds $\log N$ in high-dimensional optimization problems, the overall per-iteration complexity is dominated by the position update and diversity enhancement procedures. Accordingly, the computational complexity per iteration of PCOA is asymptotically approximated by $O(ND)$.

Over T total iterations, the cumulative computational complexity of the PCOA algorithm can therefore be expressed as $O(TND)$. This linear relationship with respect to the population size, dimensionality, and iteration count underscores the algorithm's scalability and computational viability, particularly in addressing complex, large-scale, and high-dimensional optimization tasks.

2.6 Convergence Analysis

Metaheuristic algorithms such as the proposed Pirates of the Caribbean Optimization Algorithm (PCOA) are inherently stochastic and iterative in nature. While it is generally challenging to provide deterministic convergence proofs for such algorithms due to their reliance on randomization, it is possible to analyze their convergence behavior in a probabilistic sense. This section presents a formal convergence analysis for PCOA under standard assumptions commonly used in the literature on population-based metaheuristics.

2.6.1 Assumptions

The following conditions are assumed for the convergence analysis:

1. **Bounded Search Space:** The solution space $S \subset \mathbb{R}^D$ is compact and bounded.

2. **Continuity:** The objective function $f : S \to \mathbb{R}$ is continuous and bounded from below.

3. **Existence of Global Optimum:** There exists at least one global optimum $x^* \in S$ such that $f(x^*) \leq f(x)$ for all $x \in S$.

4. **Positive Reachability:** For any $x \in S$ and any open neighborhood $U(x, \varepsilon)$, there exists a nonzero probability that a candidate solution generated by the algorithm will fall within $U(x, \varepsilon)$ in a finite number of iterations, due to the randomization mechanisms (such as Lévy flights, sinusoidal perturbation, and Gaussian quasi-reflection).

2.6.2 Probabilistic Convergence Argument

Let $P^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \ldots, x_N^{(t)}\}$ denote the population of candidate solutions (pirate ships) at iteration t, and let x_t^* represent the best solution identified up to iteration t. The convergence argument is based on two key properties of PCOA:

1. **Ergodic Search Capability:** Due to the incorporation of various stochastic operators, including Lévy flights, perturbation mechanisms, and quasi-reflection, PCOA has a nonzero probability of generating candidate solutions in any neighborhood of the global optimum x^*. That is,
$$\Pr\left(\exists t \in \mathbb{N} : x_i^{(t)} \in U(x^*, \varepsilon)\right) > 0, \quad \forall \varepsilon > 0. \tag{35}$$

2. **Elitist Retention:** The elitism protocol with rollback ensures that once a superior solution is found, it is preserved in the population or restored if a population update does not yield an improvement. This mechanism guarantees a non-increasing sequence of best objective values:
$$f\left(x_{t+1}^*\right) \leq f(x_t^*), \quad \forall t. \tag{36}$$

Given these properties and the boundedness of f, the sequence $\{f(x_t^*)\}$ forms a monotonic non-increasing and bounded stochastic process. By the monotone convergence theorem, it follows that
$$\lim_{t \to \infty} f(x_t^*) = f^*, \tag{37}$$
almost surely, where f^* is the value of the global optimum.

Furthermore, because the search process is ergodic and the best solution is always retained, the probability that the global optimum is eventually found approaches one as the number of iterations increases:
$$\lim_{t \to \infty} \Pr(\|x_t^* - x^*\| < \varepsilon) = 1, \quad \forall \varepsilon > 0. \tag{38}$$

2.6.3 Conclusion

In summary, under the stated assumptions, the PCOA algorithm converges asymptotically in probability to the global optimum. The diversity mechanisms guarantee that the search process does not stagnate prematurely, while elitist retention ensures that superior solutions are never lost. This convergence analysis, together with the empirical results, supports the reliability and robustness of PCOA in solving complex optimization problems.

2.7 Empirical Validation and Performance Assessment

To rigorously evaluate the effectiveness of the proposed PCOA algorithm in solving optimization problems, a comprehensive set of experiments was conducted across a diverse array of challenges, including classical test functions, the CEC2022 benchmark suite, and real-world scenarios such as Nonlinear Inverse Partial Differential Equations (IPDEs) [32]. The empirical results demonstrate that:

- PCOA achieves superior convergence rates compared to state-of-the-art metaheuristics.
- Its adaptive balance between exploration and exploitation enhances robustness across highly deceptive and multimodal search landscapes.
- The algorithm maintains high computational efficiency, affirming its practical applicability to complex problem domains.

To validate the statistical significance of PCOAs performance, comparisons were made against several leading metaheuristics using the CEC 2019 and classical benchmarks. The rank-sum and Friedman tests were employed, confirming the superior performance of PCOA with statistical confidence.

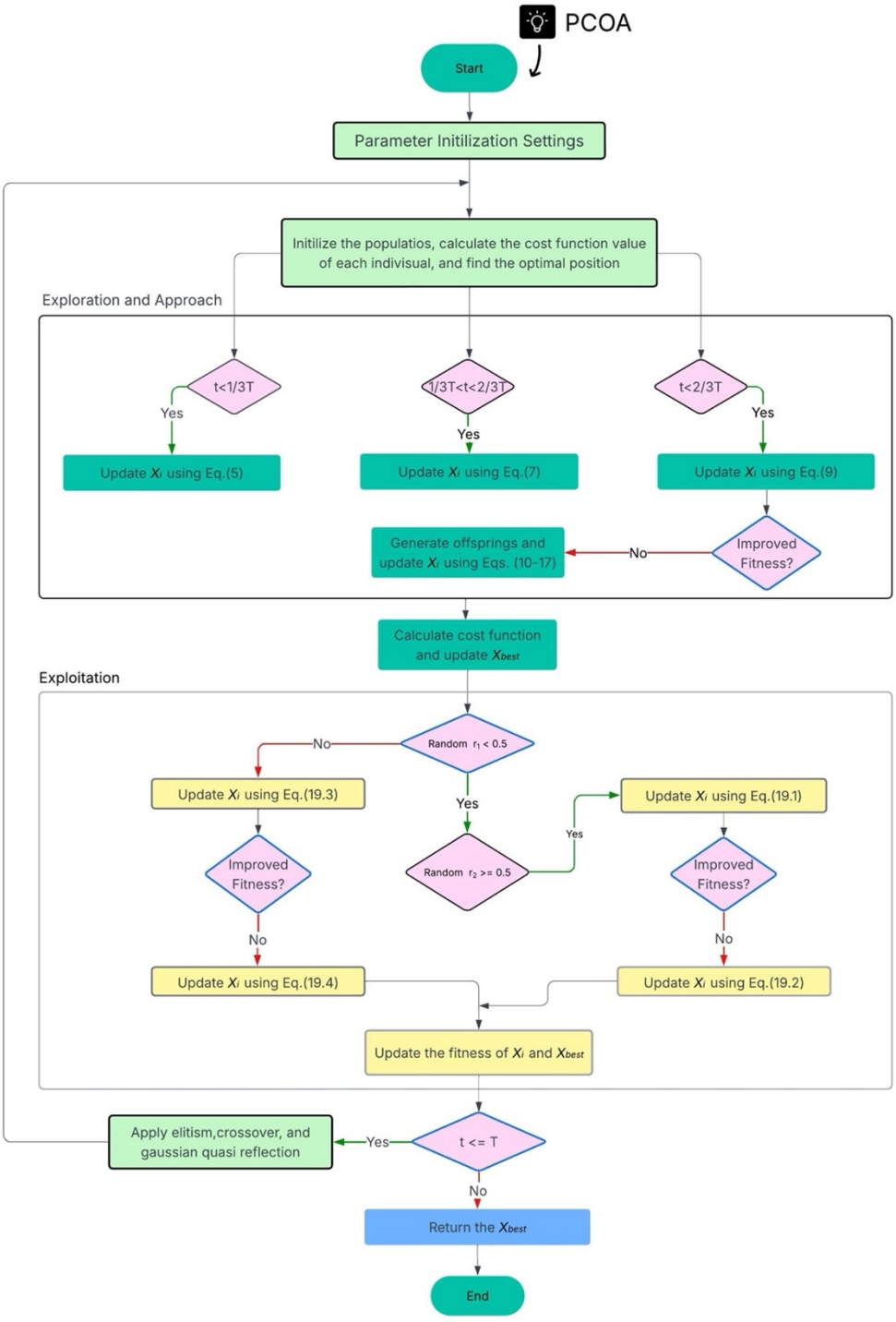

Figure 2. Flowchart of the Proposed PCOA

2.7.1 Numerical Experiments

In this section, the performance of PCOA is evaluated using two categories of test functions: twenty-three classical benchmark functions and the CEC 2017 test functions.

2.7.1.1 Results of PCOA on the Twenty-Three Classical Benchmark Functions

To thoroughly evaluate the performance of PCOA and other swarm intelligence (SI) algorithms, 23 standard benchmark functions are employed, categorized into three groups: unimodal (F1F7) for assessing convergence and local exploitation, multimodal (F8F13) for testing global exploration and avoidance of local optima, and fixed-dimension multimodal (F14F23) for measuring robustness in complex low-dimensional settings. The detailed definitions and characteristics of these functions are provided in Table 1.
To evaluate the performance of the proposed PCOA, several well-established and recently introduced metaheuristic algorithms were selected for comparison. The benchmark set includes six classic algorithms: Genetic Algorithm (GA) [7], Differential Evolution (DE) [33], Particle Swarm Optimization (PSO) [9], Ant Colony Optimization (ACO) [8], Artificial Bee Colony (ABC) [34], and Bat Algorithm (BA) [35]. In addition, newer or nature-inspired methods were also considered, including the Whale Optimization Algorithm (WOA) [36], Grey Wolf Optimizer (GWO) [37], Secretary Bird Optimization Algorithm (SBOA) [38], and Grasshopper Optimization Algorithm (GOA) [39].

All comparative algorithms were evaluated on the same 23 classical benchmark functions and CEC-2022 benchmark functions as originally defined in their respective publications, ensuring fairness and reproducibility. This comprehensive comparative analysis serves to validate the experimental robustness and general effectiveness of PCOA in handling diverse optimization landscapes.

To comprehensively evaluate the performance of the proposed PCOA algorithm, experiments were conducted on two benchmark sets: (1) a collection of 23 classical benchmark functions, and (2) the CEC 2017 benchmark suite, comprising complex, shifted, rotated, hybrid, and composition functions widely recognized for testing the robustness and generalization capabilities of optimization algorithms. For all experiments, the population size was set to 60, and the maximum number of iterations was fixed at 1000, resulting in a total of 60,000 fitness evaluations per run for each benchmark function. Each algorithm was independently executed 30 times on every test function to account for stochastic variability and ensure the statistical robustness of the results.

To compare the performance of PCOA with other algorithms, both descriptive statistics (including best, mean, worst, median and standard deviation) and non-parametric statistical tests were used. The Wilcoxon signed-rank test was applied for pairwise comparison between PCOA and each competitor, while the Friedman test was employed to evaluate overall ranking differences among all algorithms. A significance level of 0.05 was adopted for all statistical tests. These methods provide a rigorous foundation for determining whether performance differences are statistically significant across both benchmark categories.

The detailed results of PCOA and ten comparative algorithms on the twenty-three classical benchmark functions along with the overall Friedman ranking are presented in Table 2. For each function, the mean, best, worst, median, and standard deviation (STD) values over 30 independent runs are reported. PCOA consistently demonstrates dominant performance, achieving the best mean solution in 21 out of 23 functions (approximately 93%) and the lowest standard deviation in 19 functions (approximately 83%), highlighting its robustness, stability, and high-precision convergence.

For the unimodal test functions (F1–F7), which evaluate the algorithms exploitation capability, PCOA achieves the best mean in all functions. This strong performance signifies PCOAs superior convergence speed and local search efficiency compared to the other algorithms.

On the multimodal functions (F8–F13), which challenge the exploration ability due to numerous local optima, PCOA outperforms all other algorithms by achieving the best mean in all 6 cases. These results clearly illustrate the algorithms exceptional global search capacity and its ability to avoid premature convergence.

The Friedman ranking test further supports these findings by ranking PCOA first overall, based on its average rank across all test functions. Additionally, the Wilcoxon rank-sum test, summarized in Table 3, was applied at a significance level of 0.05. In this test, a "$+$" indicates that PCOA performs significantly better than a comparator, "$\sim$" indicates no significant difference, and "$-$" indicates a loss. The outcomes confirm that PCOA has statistically significant superiority over most competing algorithms.

Moreover, the convergence behavior of the algorithms is visualized in Figure 3, where PCOA shows rapid convergence on functions such as F1–F4, F9–F11, F18, and F20. Although PCOA converges more slowly at early stages in F6, F12, and F13, it consistently reaches superior final solutions. This pattern demonstrates PCOAs strong capability to escape local optima and reinforces its overall reliability and

Table 1. Twenty-three classical benchmark functions

Function	Dimension	Range	$f_{\min}$
$F_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]$	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]$	0
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]$	0
$F_4(x)=\max_{1\le i\le n}\lvert x_i\rvert$	30	$[-100,100]$	0
$F_5(x)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	30	$[-30,30]$	0
$F_6(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	30	$[-100,100]$	0
$F_7(x)=\sum_{i=1}^{n}ix_i^4+\mathrm{rand}(0,1)$	30	$[-1.28,1.28]$	0
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]$	$-418.9829n$
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	$[-5.12,5.12]$	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum x_i^2}\right)-\exp\left(\frac{1}{n}\sum\cos(2\pi x_i)\right)+20+e$	30	$[-32,32]$	0
$F_{11}(x)=\frac{1}{4000}\sum x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]$	0
$F_{12}(x)=\frac{\pi}{n}\left[10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2(1+10\sin^2(\pi y_{i+1}))+(y_n-1)^2\right]+\sum u(x_i,10,100,4)$	30	$[-50,50]$	0
$F_{13}(x)=0.1\left[\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2(1+\sin^2(3\pi x_{i+1}))+(x_n-1)^2(1+\sin^2(2\pi x_n))\right]+\sum u(x_i,5,100,4)$	30	$[-50,50]$	0
$F_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65.536,65.536]$	1
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.0003075
$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{x_1^6}{3}+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]$	-1.0316
$F_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$[-5,10]\times[0,15]$	0.3979
$F_{18}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]$ $\times\left[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	$[-2,2]$	3
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	3	$[0,1]$	-3.86
$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	6	$[0,1]$	-3.32
$F_{21}(x)=-\sum_{i=1}^{5}\left[\sum_{j=1}^{4}(x_j-a_{ij})^2+c_i\right]^{-1}$	4	$[0,10]$	-10.1532
$F_{22}(x)=-\sum_{i=1}^{7}\left[\sum_{j=1}^{4}(x_j-a_{ij})^2+c_i\right]^{-1}$	4	$[0,10]$	-10.4029
$F_{23}(x)=-\sum_{i=1}^{10}\left[\sum_{j=1}^{4}(x_j-a_{ij})^2+c_i\right]^{-1}$	4	$[0,10]$	-10.5364

effectiveness across various types of fitness landscapes.

2.7.1.2 Results of PCOA on the CEC 2019 Benchmark Functions

To further validate the effectiveness of the proposed PCOA algorithm, the CEC 2019 benchmark suite was adopted, which contains a variety of complex, shifted, rotated, and composition functions designed to challenge both exploration and exploitation capabilities of optimization algorithms. The experimental setup, including the maximum number of iterations, population size, parameter values, and runtime configuration, was kept consistent with the previous experiments for fair comparison. PCOA was evaluated against ten competitive algorithms: WOA, GWO, BA, DE, GA, ABC, ACO, PSO, SBOA, and GOA. Details of the CEC 2019 test suite can be found in Ref. [40].

According to the Friedman ranking test, PCOA achieved a ranking score of 3.1, placing it first among all competitors, followed by SBOA with a score of 3.65. Additionally, PCOA achieved the best average optimal solutions on 9 out of 10 functions (F1–F10), with the exception of F7, yielding a success rate of 90%. The slight underperformance on F7 can be attributed to the function's complex structure, which likely requires more delicate balance between exploration and exploitation than other functions in the suite. The comparative results between PCOA and the other algorithms, along with the Friedman ranking test outcomes, are reported in Table 4.

The Wilcoxon rank-sum test results, summarized in Table 4, confirm that PCOA statistically outperforms comparative algorithms on more than 90% of the CEC 2019 functions. In particular, PCOA demonstrates significantly better performance compared to WOA, GWO, and SBOA, suggesting its superior robustness and adaptability in high-complexity landscapes.

Furthermore, Figure 4 illustrates the convergence curves of PCOA and other algorithms across the benchmark suite. It is evident that PCOA exhibits the fastest convergence speed on functions F1-F10, and maintains a competitive convergence profile on the remaining functions. This pattern aligns with the Friedman ranking and further demonstrates PCOA's ability to reach very small objective values in few iterations.

Table 2. Comparison of PCOA with ten Metaheuristic Algorithms on Twenty-Three Classical Benchmark Functions

F	Measure	PCOA	WOA	GWO	BA	DE	GA	ABC	ACO	PSO	SBOA	GOA
F1	Mean	**0.000E+00**	5.474E-208	1.267E-80	6.301E+04	2.024E+03	2.351E+03	6.464E+03	4.915E+03	5.515E+03	**0.000E+00**	1.750E-02
	Best	**0.000E+00**	6.877E-224	4.838E-83	4.874E+04	4.746E+02	1.696E+03	4.882E+03	3.037E+03	2.213E+03	**0.000E+00**	4.955E-03
	Worst	**0.000E+00**	1.638E-206	1.207E-79	7.333E+04	4.806E+03	2.771E+03	8.269E+03	8.131E+03	1.031E+04	**0.000E+00**	5.567E-02
	Median	**0.000E+00**	7.011E-214	4.310E-81	6.262E+04	1.907E+03	2.406E+03	6.502E+03	4.773E+03	5.086E+03	**0.000E+00**	1.490E-02
	STD	**0.000E+00**	0.000E+00	2.586E-80	5.705E+03	1.002E+03	2.614E+02	7.735E+02	1.192E+03	1.913E+03	**0.000E+00**	1.085E-02
F2	Mean	**0.000E+00**	6.867E-129	7.817E-47	1.113E+11	6.634E+00	9.774E+00	4.499E+01	6.647E+01	4.350E+01	8.518E-210	3.272E-03
	Best	**0.000E+00**	2.400E-137	5.636E-48	1.592E+08	2.559E+00	7.601E+00	3.208E+01	5.123E+01	2.168E+01	4.111E-225	2.157E-03
	Worst	**0.000E+00**	7.606E-128	3.017E-46	1.392E+12	1.548E+01	1.195E+01	5.162E+01	1.190E+02	7.676E+01	2.009E-208	5.002E-03
	Median	**0.000E+00**	4.372E-131	5.455E-47	1.228E+10	5.481E+00	9.809E+00	4.608E+01	6.338E+01	3.715E+01	1.505E-216	3.131E-03
	STD	**0.000E+00**	1.746E-128	7.025E-47	2.564E+11	3.675E+00	8.579E-01	4.299E+00	1.331E+01	1.530E+01	**0.000E+00**	5.941E-04
F3	Mean	**0.000E+00**	1.667E+04	4.082E-14	1.025E+05	8.844E+03	2.246E+04	4.014E+04	6.584E+04	9.391E+03	6.156E-266	2.341E+02
	Best	**0.000E+00**	4.453E+03	5.628E-20	5.881E+04	3.863E+03	1.736E+04	3.107E+04	4.767E+04	4.252E+03	5.716E-305	5.042E+00
	Worst	**0.000E+00**	2.890E+04	9.543E-13	1.562E+05	2.220E+04	2.965E+04	4.592E+04	7.906E+04	1.733E+04	1.782E-264	5.042E+03
	Median	**0.000E+00**	1.620E+04	1.353E-15	1.022E+05	8.017E+03	2.191E+04	4.088E+04	6.605E+04	9.331E+03	6.358E-279	2.051E+01
	STD	**0.000E+00**	5.810E+03	1.714E-13	2.180E+04	3.585E+03	2.977E+03	3.255E+03	7.269E+03	2.919E+03	**0.000E+00**	9.257E+02
F4	Mean	**0.000E+00**	8.045E+01	1.443E-15	8.395E+01	4.278E+01	3.341E+01	6.816E+01	8.429E+01	3.668E+01	5.837E-171	7.828E-01
	Best	**0.000E+00**	5.919E+01	1.871E-17	7.330E+01	2.603E+01	2.956E+01	5.720E+01	7.511E+01	2.459E+01	1.060E-187	2.908E-01
	Worst	**0.000E+00**	9.058E+01	9.697E-15	9.038E+01	6.621E+01	3.717E+01	7.375E+01	9.333E+01	4.811E+01	1.484E-169	1.709E+00
	Median	**0.000E+00**	8.131E+01	8.090E-16	8.521E+01	4.351E+01	3.346E+01	6.890E+01	8.454E+01	3.637E+01	2.300E-181	7.142E-01
	STD	**0.000E+00**	6.463E+00	2.044E-15	4.506E+00	7.520E+00	1.548E+00	3.712E+00	4.012E+00	6.595E+00	**0.000E+00**	3.097E-01
F5	Mean	**0.000E+00**	2.582E+01	2.565E+01	2.235E+08	1.427E+06	1.177E+06	1.405E+07	1.293E+07	4.095E+06	2.403E+01	1.074E+02
	Best	**0.000E+00**	2.436E+01	2.500E+01	1.437E+08	4.889E+05	8.269E+05	4.894E+06	6.381E+06	5.006E+05	2.367E+01	2.593E+01
	Worst	**0.000E+00**	2.830E+01	2.706E+01	3.113E+08	5.821E+06	1.561E+06	2.135E+07	2.007E+07	1.412E+07	2.500E+01	5.631E+02
	Median	**0.000E+00**	2.597E+01	2.546E+01	2.234E+08	1.110E+06	1.206E+06	1.438E+07	1.219E+07	4.073E+06	2.398E+01	2.911E+01
	STD	**0.000E+00**	8.067E-01	5.027E-01	3.761E+07	1.134E+06	2.042E+05	3.758E+06	4.106E+06	2.859E+06	2.924E-01	1.257E+02
F6	Mean	**2.237E-15**	4.506E-04	5.986E-02	6.489E+04	1.566E+03	2.329E+03	6.064E+03	5.242E+03	5.603E+03	5.154E-08	1.811E-02
	Best	**1.059E-15**	7.179E-06	2.337E-06	4.671E+04	3.747E+02	1.802E+03	4.202E+03	4.084E+03	3.299E+03	2.239E-08	5.456E-03
	Worst	**3.823E-15**	3.251E-03	2.551E-01	7.337E+04	2.875E+03	2.916E+03	7.134E+03	7.123E+03	8.458E+03	8.087E-08	3.439E-02
	Median	**2.197E-15**	2.323E-04	6.255E-06	6.629E+04	1.818E+03	2.349E+03	6.159E+03	5.195E+03	5.248E+03	5.325E-08	1.683E-02
	STD	**6.688E-16**	6.256E-04	1.027E-01	6.642E+03	8.163E+02	2.839E+02	7.137E+02	8.524E+02	1.476E+03	1.847E-08	7.478E-03
F7	Mean	**3.397E-06**	1.248E-03	7.633E-04	1.095E+02	6.132E-01	8.152E-01	6.372E+00	4.828E+00	1.646E+00	1.258E-04	2.693E-02
	Best	**6.574E-08**	2.767E-05	2.750E-04	6.338E+01	5.302E-02	5.852E-01	2.804E+00	2.867E+00	4.873E-01	1.866E-05	9.620E-03
	Worst	**1.309E-05**	7.972E-03	2.981E-03	1.489E+02	1.502E+00	1.031E+00	1.030E+01	7.212E+00	4.622E+00	3.014E-04	6.342E-02
	Median	**2.712E-06**	6.797E-04	6.521E-04	1.127E+02	5.042E-01	8.355E-01	6.551E+00	5.012E+00	1.406E+00	1.055E-04	2.518E-02
	STD	**3.109E-06**	1.959E-03	4.913E-04	2.075E+01	4.203E-01	1.150E-01	1.609E+00	1.185E+00	8.448E-01	8.037E-05	1.287E-02
F8	Mean	**-1.235E+04**	-8.766E+03	-6.492E+03	-2.781E+03	-1.149E+04	-1.204E+04	-4.656E+03	-4.384E+03	-6.190E+03	-9.099E+03	-9.379E+03
	Best	**-1.257E+04**	**-1.257E+04**	-8.470E+03	-3.441E+03	-1.229E+04	-1.229E+04	-5.526E+03	-4.953E+03	-7.347E+03	-1.048E+04	-1.066E+04
	Worst	-1.043E+04	-7.415E+03	-3.254E+03	-2.069E+03	-1.069E+04	**-1.184E+04**	-4.303E+03	-3.981E+03	-4.904E+03	-7.827E+03	-8.518E+03
	Median	**-1.251E+04**	-8.770E+03	-6.965E+03	-2.782E+03	-1.147E+04	-1.204E+04	-4.600E+03	-4.390E+03	-6.273E+03	-9.101E+03	-9.591E+03
	STD	4.205E+02	8.063E+02	1.520E+03	3.730E+02	4.358E+02	9.872E+01	**2.559E+02**	2.633E+02	5.686E+02	6.599E+02	5.302E+02
F9	Mean	**0.000E+00**	7.084E+01	1.010E+01	4.259E+02	2.286E+01	3.832E+01	2.722E+02	2.975E+02	1.095E+02	**0.000E+00**	5.761E+01
	Best	**0.000E+00**	**0.000E+00**	**0.000E+00**	3.866E+02	1.166E+01	3.342E+01	2.268E+02	2.650E+02	6.771E+01	**0.000E+00**	2.686E+01
	Worst	**0.000E+00**	2.272E+02	5.099E+01	4.592E+02	3.333E+01	4.706E+01	2.963E+02	3.211E+02	1.651E+02	**0.000E+00**	9.353E+01
	Median	**0.000E+00**	6.415E+01	6.087E+00	4.246E+02	2.479E+01	3.734E+01	2.744E+02	2.991E+02	1.106E+02	**0.000E+00**	6.169E+01
	STD	**0.000E+00**	7.263E+01	1.104E+01	2.012E+01	6.231E+00	3.295E+00	1.283E+01	1.304E+01	2.146E+01	**0.000E+00**	1.664E+01
F10	Mean	**4.441E-16**	5.063E-15	6.721E-15	2.001E+01	8.578E+00	1.012E+01	1.634E+01	2.022E+01	1.423E+01	**4.441E-16**	6.612E-01
	Best	**4.441E-16**	**4.441E-16**	3.997E-15	1.991E+01	5.640E+00	9.016E+00	1.521E+01	2.002E+01	1.136E+01	**4.441E-16**	1.808E-03
	Worst	**4.441E-16**	1.465E-14	7.550E-15	2.010E+01	1.152E+01	1.094E+01	1.697E+01	2.034E+01	1.785E+01	**4.441E-16**	1.970E+01
	Median	**4.441E-16**	3.997E-15	7.550E-15	2.000E+01	8.429E+00	1.017E+01	1.644E+01	2.025E+01	1.441E+01	**4.441E-16**	3.857E-03
	STD	**0.000E+00**	2.922E-15	1.503E-15	4.271E-02	1.450E+00	4.128E-01	4.496E-01	7.398E-02	1.820E+00	**0.000E+00**	3.535E+00
F11	Mean	**0.000E+00**	**0.000E+00**	2.154E-03	5.606E+02	1.680E+01	2.226E+01	5.890E+01	4.590E+01	5.645E+01	**0.000E+00**	3.043E-02
	Best	**0.000E+00**	**0.000E+00**	**0.000E+00**	4.305E+02	7.262E+00	1.617E+01	4.301E+01	3.278E+01	2.496E+01	**0.000E+00**	5.666E-03
	Worst	**0.000E+00**	**0.000E+00**	3.055E-02	6.175E+02	4.625E+01	2.636E+01	7.552E+01	6.196E+01	1.104E+02	**0.000E+00**	7.784E-02
	Median	**0.000E+00**	**0.000E+00**	**0.000E+00**	5.652E+02	1.416E+01	2.277E+01	5.807E+01	4.515E+01	5.284E+01	**0.000E+00**	2.361E-02
	STD	**0.000E+00**	**0.000E+00**	6.958E-03	4.200E+01	8.622E+00	2.659E+00	9.191E+00	5.956E+00	1.830E+01	**0.000E+00**	2.099E-02
F12	Mean	**2.473E-16**	5.511E-01	3.952E-03	4.895E+08	3.319E+05	9.729E+04	1.980E+07	2.783E+07	5.470E+05	5.961E-09	2.420E-02
	Best	**6.916E-17**	1.780E-05	3.470E-07	2.915E+08	2.645E+03	2.165E+04	5.306E+06	1.068E+07	5.275E+01	1.902E-09	1.626E-06
	Worst	**5.690E-16**	7.086E+00	1.937E-02	6.929E+08	3.007E+06	2.050E+05	3.731E+07	6.552E+07	3.389E+06	9.939E-09	2.073E-01
	Median	**2.498E-16**	7.394E-04	1.074E-06	4.616E+08	1.256E+05	9.622E+04	1.897E+07	2.498E+07	1.376E+05	5.980E-09	2.037E-05
	STD	**1.182E-16**	1.526E+00	5.239E-03	1.053E+08	6.158E+05	5.039E+04	7.516E+06	1.182E+07	8.336E+05	1.879E-09	5.137E-02
F13	Mean	**1.350E-32**	4.670E-01	7.402E-02	9.835E+08	3.624E+06	1.923E+06	5.396E+07	6.454E+07	5.436E+06	4.650E-02	3.522E-03
	Best	**1.350E-32**	2.005E-02	4.686E-06	6.087E+08	1.183E+04	7.838E+05	1.567E+07	2.745E+07	1.784E+05	3.155E-08	8.739E-06
	Worst	**1.350E-32**	5.158E+00	4.923E-01	1.298E+09	1.844E+07	3.104E+06	9.618E+07	1.216E+08	1.936E+07	2.123E-01	1.201E-02
	Median	**1.350E-32**	2.842E-01	1.743E-05	9.960E+08	1.914E+06	1.875E+06	5.237E+07	6.295E+07	5.480E+06	1.784E-07	1.970E-04
	STD	**2.737E-48**	9.096E-01	1.064E-01	1.594E+08	4.404E+06	5.597E+05	1.686E+07	2.335E+07	4.362E+06	6.154E-02	5.103E-03
F14	Mean	2.148E+00	1.031E+00	1.064E+00	1.093E+01	2.147E+00	**9.980E-01**	**9.980E-01**	1.041E+00	3.417E+00	**9.980E-01**	**9.980E-01**
	Best	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**
	Worst	1.076E+01	1.992E+00	2.982E+00	2.644E+01	1.076E+01	**9.980E-01**	**9.980E-01**	1.805E+00	1.644E+01	**9.980E-01**	**9.980E-01**
	Median	**9.980E-01**	**9.980E-01**	**9.980E-01**	8.788E+00	**9.980E-01**	**9.980E-01**	**9.980E-01**	**9.980E-01**	2.982E+00	**9.980E-01**	**9.980E-01**
	STD	2.117E+00	1.784E-01	3.562E-01	6.696E+00	2.295E+00	1.189E-08	9.271E-08	1.499E-01	3.542E+00	**0.000E+00**	1.216E-16
F15	Mean	**3.075E-04**	5.423E-04	3.116E-03	2.021E-02	1.598E-03	2.157E-03	7.387E-04	2.642E-03	6.319E-04	1.037E-03	1.190E-03
	Best	**3.075E-04**	**3.075E-04**	**3.075E-04**	1.972E-03	5.133E-04	7.519E-04	4.854E-04	8.447E-04	**3.075E-04**	**3.075E-04**	**3.075E-04**
	Worst	**3.075E-04**	2.252E-03	2.036E-02	6.983E-02	7.658E-03	2.043E-02	8.648E-04	1.258E-02	1.594E-03	2.036E-02	2.036E-02
	Median	**3.075E-04**	**3.075E-04**	**3.075E-04**	1.110E-02	9.367E-04	1.526E-03	7.553E-04	1.130E-03	**3.075E-04**	**3.075E-04**	**3.075E-04**
	STD	**3.882E-18**	5.476E-04	6.774E-03	1.949E-02	1.511E-03	3.426E-03	6.434E-05	3.121E-03	4.199E-04	3.596E-03	3.581E-03
F16	Mean	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	-7.728E-01	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**
	Best	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	-1.025E+00	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**
	Worst	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	-3.116E-01	-1.032E+00	-1.031E+00	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**
	Median	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	-8.109E-01	-1.032E+00	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**
	STD	4.441E-16	**3.511E-16**	6.139E-10	2.151E-01	1.139E-05	3.218E-05	4.441E-16	4.441E-16	4.328E-16	4.441E-16	3.695E-13
F17	Mean	**3.979E-01**	**3.979E-01**	**3.979E-01**	8.160E-01	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**
	Best	**3.979E-01**	**3.979E-01**	**3.979E-01**	3.990E-01	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**
	Worst	**3.979E-01**	**3.979E-01**	3.981E-01	1.871E+00	3.983E-01	**3.980E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	3.979E-01
	Median	**3.979E-01**	**3.979E-01**	**3.979E-01**	6.159E-01	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	**3.979E-01**	3.979E-01
	STD	**0.000E+00**	3.189E-16	3.959E-05	4.492E-01	7.366E-05	2.217E-05	**0.000E+00**	9.159E-08	**0.000E+00**	**0.000E+00**	1.165E-13
F18	Mean	**3.000E+00**	**3.000E+00**	**3.000E+00**	1.233E+01	4.861E+00	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**
	Best	**3.000E+00**	**3.000E+00**	**3.000E+00**	3.001E+00	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**
	Worst	**3.000E+00**	**3.000E+00**	**3.000E+00**	9.061E+01	3.004E+01	3.001E+00	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**
	Median	**3.000E+00**	**3.000E+00**	**3.000E+00**	6.529E+00	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**	**3.000E+00**
	STD	1.320E-15	2.316E-15	9.603E-07	1.647E+01	5.585E+00	2.123E-04	**8.542E-16**	9.628E-16	1.354E-15	1.307E-15	3.127E-12
F19	Mean	**-3.863E+00**	-3.837E+00	**-3.863E+00**	-3.619E+00	-3.862E+00	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**
	Best	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	-3.862E+00	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**
	Worst	**-3.863E+00**	-3.090E+00	-3.861E+00	-3.208E+00	-3.850E+00	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**
	Median	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	-3.626E+00	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**	**-3.863E+00**
	STD	**3.739E-16**	1.388E-01	2.815E-04	1.698E-01	2.347E-03	2.721E-06	8.882E-16	8.882E-16	8.426E-16	8.882E-16	3.819E-14
F20	Mean	-3.314E+00	-3.267E+00	-3.262E+00	-1.948E+00	-3.256E+00	-3.282E+00	**-3.322E+00**	-3.310E+00	-3.261E+00	-3.286E+00	-3.251E+00
	Best	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	-2.901E+00	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**
	Worst	-3.203E+00	-3.203E+00	-3.138E+00	-1.116E+00	-3.159E+00	-3.203E+00	**-3.322E+00**	-3.203E+00	-3.172E+00	-3.203E+00	-3.203E+00
	Median	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	-1.970E+00	-3.247E+00	**-3.322E+00**	**-3.322E+00**	**-3.322E+00**	-3.263E+00	**-3.322E+00**	-3.203E+00
	STD	**2.116E-02**	5.931E-02	6.579E-02	4.920E-01	6.026E-02	5.605E-02	5.224E-06	3.567E-02	6.158E-02	5.448E-02	5.825E-02
F21	Mean	**-1.015E+01**	-7.297E+00	-8.806E+00	-1.078E+00	-7.569E+00	-5.490E+00	-9.565E+00	-5.529E+00	-5.727E+00	**-1.015E+01**	-7.640E+00
	Best	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	-2.727E+00	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**
	Worst	**-1.015E+01**	-2.630E+00	-5.101E+00	-3.794E-01	-2.629E+00	-2.630E+00	-7.332E+00	-2.630E+00	-2.630E+00	**-1.015E+01**	-2.630E+00
	Median	**-1.015E+01**	**-1.015E+01**	**-1.015E+01**	-9.430E-01	-1.014E+01	-2.683E+00	-9.952E+00	-3.487E+00	-3.869E+00	**-1.015E+01**	**-1.015E+01**
	STD	**3.178E-15**	3.357E+00	2.234E+00	5.902E-01	3.148E+00	3.574E+00	7.394E-01	3.357E+00	3.459E+00	2.333E-12	2.968E+00
F22	Mean	**-1.040E+01**	-7.393E+00	**-1.040E+01**	-1.052E+00	-6.993E+00	-6.218E+00	-1.037E+01	-9.892E+00	-6.514E+00	**-1.040E+01**	-9.442E+00
	Best	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	-4.188E+00	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**
	Worst	**-1.040E+01**	-1.838E+00	**-1.040E+01**	-4.698E-01	-2.678E+00	-1.838E+00	-9.967E+00	-2.766E+00	-2.752E+00	**-1.040E+01**	-2.752E+00
	Median	**-1.040E+01**	**-1.040E+01**	**-1.040E+01**	-8.474E-01	-7.327E+00	-3.724E+00	**-1.040E+01**	**-1.040E+01**	-5.108E+00	**-1.040E+01**	**-1.040E+01**
	STD	**3.370E-15**	3.525E+00	3.239E-05	7.431E-01	3.351E+00	3.711E+00	9.798E-02	1.713E+00	3.475E+00	5.983E-11	2.183E+00
F23	Mean	**-1.054E+01**	-7.532E+00	-1.036E+01	-1.081E+00	-8.397E+00	-7.327E+00	-1.051E+01	-1.027E+01	-7.820E+00	**-1.054E+01**	-8.769E+00
	Best	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	-3.981E+00	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**
	Worst	**-1.054E+01**	-1.677E+00	-5.128E+00	-5.604E-01	-2.803E+00	-2.422E+00	-9.716E+00	-2.427E+00	-1.859E+00	**-1.054E+01**	-2.807E+00
	Median	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	-9.007E-01	-1.052E+01	-1.053E+01	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**	**-1.054E+01**
	STD	**3.508E-15**	3.325E+00	9.707E-01	6.132E-01	3.102E+00	3.460E+00	1.473E-01	1.456E+00	3.611E+00	3.069E-14	2.751E+00
Friedman	Average Ranking	3.108695652	4.52173913	4.369565217	10.69565217	5.760869565	6.586956522	7.652173913	8.217391304	6.413043478	3.652173913	5.02173913
	Ranking	1	4	3	11	6	8	9	10	7	2	5

Table 3. The Wilcoxon rank-sum test results between PCOA and ten comparative algorithms on classical benchmark functions

F	WOA p (Sig.)	GWO p (Sig.)	BA p (Sig.)	DE p (Sig.)	GA p (Sig.)	ABC p (Sig.)	ACO p (Sig.)	PSO p (Sig.)	SBOA p (Sig.)	GOA p (Sig.)
F1	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11
F2	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F3	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F4	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F5	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F6	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F7	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F8	(+) 1.54E-10	(+) 2.87E-11	(+) 2.87E-11	(+) 1.12E-09	(+) 2.58E-06	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F9	(+) 1.63E-04	(+) 2.95E-08	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11
F10	(+) 8.12E-09	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11
F11	(~) 1.00E+00	(~) 5.06E-01	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11
F12	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F13	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F14	(~) 5.01E-02	(~) 1.47E-01	(+) 2.79E-09	(~) 3.87E-01	(~) 1.83E-01	(~) 2.31E-01	(~) 5.95E-01	(+) 6.38E-03	(-) 7.79E-03	(~) 3.52E-01
F15	(+) 5.84E-10	(+) 8.56E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 4.29E-01	(+) 4.01E-10	(+) 2.49E-10
F16	(-) 8.79E-04	(+) 2.87E-11	(+) 2.87E-11	(~) 1.83E-01	(+) 2.87E-11	(~) 1.00E+00	(~) 1.00E+00	(~) 6.57E-01	(~) 1.00E+00	(+) 2.87E-11
F17	(~) 8.24E-01	(+) 2.87E-11	(+) 2.87E-11	(+) 9.19E-06	(+) 2.87E-11	(~) 1.00E+00	(~) 8.24E-01	(~) 1.00E+00	(~) 1.00E+00	(+) 2.87E-11
F18	(+) 3.27E-02	(+) 2.87E-11	(+) 2.87E-11	(+) 7.89E-07	(+) 2.87E-11	(+) 3.39E-02	(~) 4.20E-01	(+) 1.27E-03	(~) 8.13E-01	(+) 2.87E-11
F19	(+) 3.39E-07	(+) 2.87E-11	(+) 2.87E-11	(+) 1.27E-10	(+) 2.87E-11	(~) 1.00E+00	(~) 1.00E+00	(~) 3.75E-01	(~) 1.00E+00	(+) 2.87E-11
F20	(+) 4.49E-08	(+) 4.49E-08	(+) 2.87E-11	(+) 2.95E-08	(+) 2.28E-07	(~) 7.79E-01	(~) 8.77E-01	(+) 2.68E-04	(+) 7.96E-03	(+) 8.12E-09
F21	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 1.07E-06	(+) 3.39E-07	(+) 1.02E-07	(+) 2.87E-11
F22	(+) 4.73E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 6.52E-09	(+) 6.05E-07	(+) 1.25E-07	(+) 8.84E-07	(+) 2.87E-11
F23	(+) 1.09E-10	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 8.29E-06	(~) 9.63E-02	(+) 5.41E-04	(+) 1.47E-02	(+) 2.87E-11
[W | T | L]	[19 | 3 | 1]	[21 | 2 | 0]	[23 | 0 | 0]	[21 | 2 | 0]	[22 | 1 | 0]	[18 | 5 | 0]	[16 | 7 | 0]	[19 | 4 | 0]	[14 | 8 | 1]	[22 | 1 | 0]

In summary, the results on the CEC 2019 test suite confirm that PCOA is highly effective for solving various complex optimization problems, combining rapid convergence with robust global search behavior.

3 Inverse Boundary Identification for Fisher and Huxley Systems Inspired by IHCPs

The process of solving nonlinear partial differential equations (PDEs) such as the Fisher and Huxley equations typically involves finding the unknown solution function $u(x,t)$ given complete knowledge of system parameters, coefficients, boundary, and initial conditions. However, in many real-world applications, critical components, such as boundary conditions, may be unknown or difficult to measure directly. This leads to the formulation of inverse problems, where missing information is inferred from observed data. This study focuses on the inverse form of the Fisher and Huxley equations, where the time-dependent boundary condition $q(t)$ is unknown. Such inverse formulations are conceptually and structurally related to inverse heat conduction problems (IHCPs), in which the boundary or initial thermal profile is estimated from transient measurements. Like IHCPs, these problems are typically ill-posed, meaning small perturbations in the data can cause large errors in the solution. Therefore, solving them requires careful mathematical treatment and the use of robust computational techniques.

In this work, the Pirates of the Caribbean Optimization Algorithm (PCOA), along with ten comparative algorithms, is employed to estimate the unknown boundary function by minimizing the discrepancy between the numerically computed PDE solution and the available observational data. This approach enables the accurate reconstruction of $q(t)$, providing insight into system dynamics even in the absence of direct boundary measurements. The methodology is applicable in various scientific and engineering contexts, including heat transfer, nonlinear wave propagation, reaction-diffusion systems, and biological modeling.

Before analyzing the inverse formulation, we first present the generalized forms of the Fisher and Huxley equations, which serve as the foundation for both the forward and inverse problem setups. These equations are defined over the spatial domain $0 \leq x \leq 1$ and time domain $t \geq 0$, and model distinct physical phenomena through nonlinear partial differential equations. The generalized form of Fisher equation is

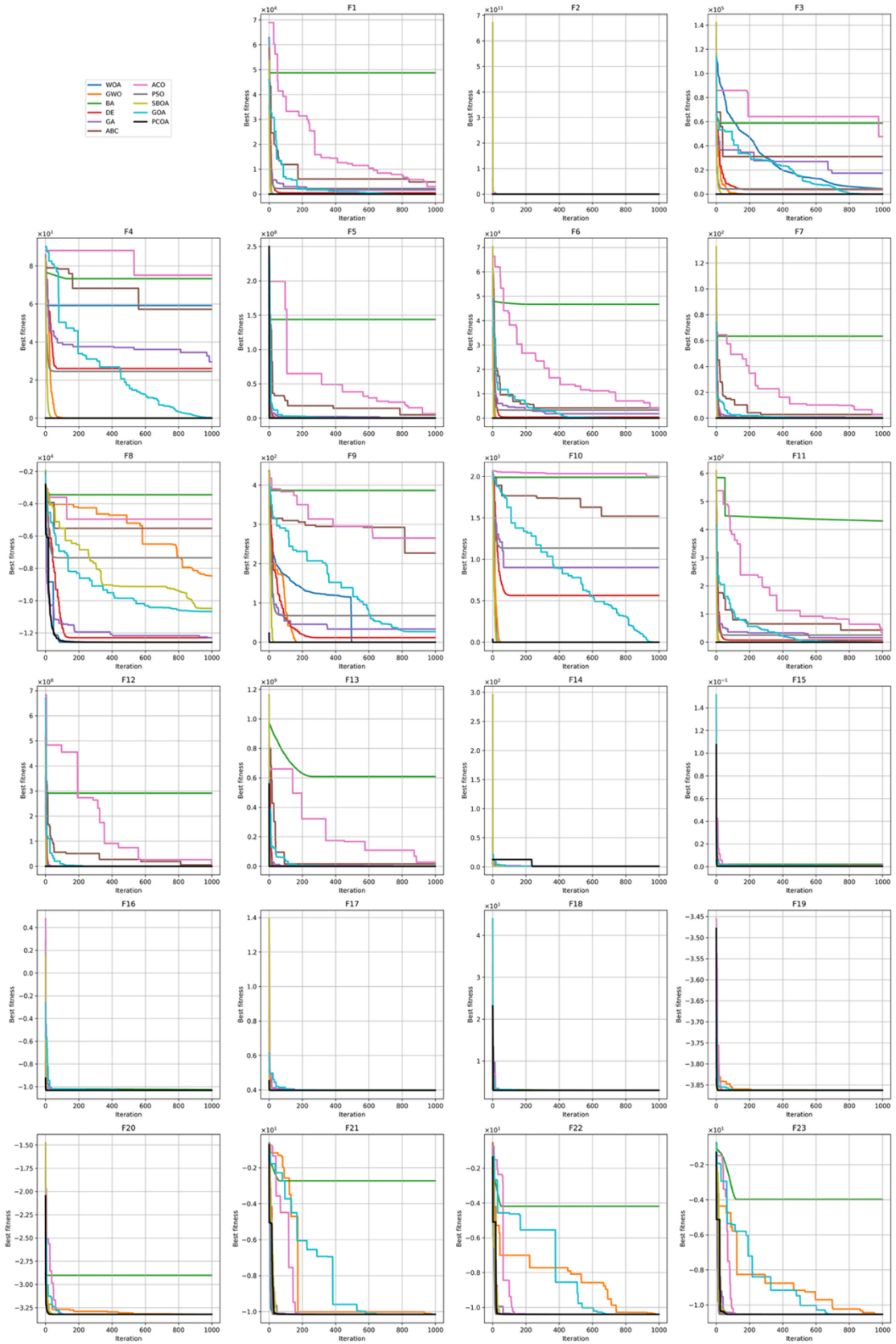

Figure 3. Convergence curves of PCOA and ten algorithms for twenty-three benchmark functions.

Table 4. Comparison of PCOA with ten Metaheuristic Algorithms on IEEE CEC 2019 Test Functions

F	Measure	PCOA	WOA	GWO	BA	DE	GA	ABC	ACO	PSO	SBOA	GOA
F1	Mean	**1.000E+00**	5.065E+06	2.939E+04	6.829E+08	9.646E+06	7.431E+06	1.283E+07	1.615E+07	4.697E+06	3.686E+02	3.815E+05
	Best	**1.000E+00**	**1.000E+00**	**1.000E+00**	1.602E+08	3.620E+05	1.029E+06	5.824E+06	3.702E+06	5.883E+03	**1.000E+00**	2.345E+04
	Worst	**1.000E+00**	1.979E+07	3.539E+05	1.634E+09	3.822E+07	1.913E+07	2.161E+07	2.888E+07	1.675E+07	1.061E+04	9.932E+05
	Median	**1.000E+00**	3.696E+06	7.211E+02	6.316E+08	5.274E+06	6.822E+06	1.259E+07	1.615E+07	2.906E+06	1.000E+00	4.058E+05
	STD	**0.000E+00**	4.682E+06	7.289E+04	3.658E+08	1.024E+07	5.057E+06	3.942E+06	5.745E+06	5.195E+06	1.903E+03	2.426E+05
F2	Mean	**4.835E+00**	3.073E+03	1.029E+02	3.543E+03	2.409E+02	4.712E+02	6.218E+02	1.038E+03	1.333E+02	6.921E+01	1.528E+02
	Best	**4.511E+00**	6.403E+02	2.794E+01	8.850E+02	6.780E+01	2.353E+02	3.243E+02	6.199E+02	1.179E+01	1.181E+01	5.688E+01
	Worst	**5.000E+00**	8.991E+03	3.185E+02	6.465E+03	4.779E+02	7.065E+02	8.887E+02	1.581E+03	4.734E+02	1.896E+02	3.518E+02
	Median	**4.946E+00**	2.490E+03	8.543E+01	3.644E+03	2.227E+02	4.695E+02	6.445E+02	1.029E+03	7.359E+01	6.395E+01	1.395E+02
	STD	**1.830E-01**	1.978E+03	6.035E+01	1.196E+03	1.003E+02	1.087E+02	1.377E+02	2.530E+02	1.196E+02	3.465E+01	7.255E+01
F3	Mean	**1.386E+00**	1.989E+00	2.385E+00	1.204E+01	3.480E+00	2.156E+00	1.010E+01	1.071E+01	6.587E+00	1.400E+00	5.877E+00
	Best	**1.000E+00**	**1.000E+00**	**1.000E+00**	1.107E+01	1.411E+00	1.612E+00	8.748E+00	8.083E+00	2.185E+00	**1.000E+00**	1.409E+00
	Worst	**2.366E+00**	5.363E+00	9.710E+00	1.270E+01	6.712E+00	3.353E+00	1.068E+01	1.151E+01	1.055E+01	2.357E+00	9.712E+00
	Median	**1.409E+00**	1.410E+00	1.416E+00	1.203E+01	3.000E+00	2.025E+00	1.014E+01	1.087E+01	6.444E+00	**1.409E+00**	6.712E+00
	STD	**2.288E-01**	1.022E+00	2.044E+00	4.841E-01	1.624E+00	4.098E-01	4.431E-01	6.462E-01	1.979E+00	3.063E-01	2.338E+00
F4	Mean	**4.947E+00**	1.260E+03	4.020E+01	1.351E+04	2.134E+02	2.552E+01	4.795E+01	4.027E+01	6.119E+01	1.023E+01	1.805E+01
	Best	**1.995E+00**	2.259E+02	1.542E+01	5.933E+03	2.398E+01	1.074E+01	3.423E+01	2.972E+01	1.095E+01	3.988E+00	6.970E+00
	Worst	**7.966E+00**	3.232E+03	7.076E+01	2.639E+04	6.511E+02	3.604E+01	5.750E+01	4.759E+01	1.532E+02	1.692E+01	3.881E+01
	Median	**4.980E+00**	1.139E+03	4.053E+01	1.285E+04	1.719E+02	2.564E+01	4.786E+01	4.061E+01	5.274E+01	9.955E+00	1.692E+01
	STD	**1.551E+00**	6.643E+02	1.585E+01	4.429E+03	1.564E+02	6.304E+00	5.152E+00	4.938E+00	3.259E+01	3.436E+00	7.519E+00
F5	Mean	**1.014E+00**	2.140E+00	1.374E+00	5.103E+00	1.168E+00	1.175E+00	1.453E+00	1.569E+00	1.192E+00	1.030E+00	1.262E+00
	Best	**1.000E+00**	1.359E+00	1.072E+00	3.370E+00	1.011E+00	1.048E+00	1.311E+00	1.438E+00	1.043E+00	**1.000E+00**	1.007E+00
	Worst	**1.044E+00**	3.467E+00	1.832E+00	6.259E+00	1.458E+00	1.371E+00	1.575E+00	1.670E+00	1.369E+00	1.108E+00	1.644E+00
	Median	**1.011E+00**	2.072E+00	1.319E+00	5.320E+00	1.170E+00	1.169E+00	1.465E+00	1.582E+00	1.195E+00	1.025E+00	1.214E+00
	STD	**1.211E-02**	5.328E-01	2.227E-01	7.159E-01	9.711E-02	8.393E-02	7.388E-02	6.496E-02	7.868E-02	2.408E-02	1.598E-01
F6	Mean	**3.023E+00**	8.862E+00	1.026E+01	1.377E+01	1.007E+01	5.144E+00	9.975E+00	1.073E+01	6.321E+00	3.060E+00	3.522E+00
	Best	**1.002E+00**	5.985E+00	8.936E+00	1.025E+01	7.164E+00	4.002E+00	8.466E+00	9.088E+00	3.446E+00	1.429E+00	1.150E+00
	Worst	5.443E+00	1.042E+01	1.105E+01	1.567E+01	1.110E+01	6.234E+00	1.073E+01	1.150E+01	9.761E+00	**4.964E+00**	7.241E+00
	Median	2.987E+00	8.904E+00	1.040E+01	1.382E+01	1.013E+01	5.194E+00	1.005E+01	1.081E+01	6.277E+00	**2.981E+00**	3.423E+00
	STD	1.110E+00	1.098E+00	5.459E-01	1.177E+00	7.103E-01	5.265E-01	**4.844E-01**	6.197E-01	1.479E+00	9.378E-01	1.393E+00
F7	Mean	1.153E+01	2.672E+02	6.929E+00	1.008E+03	2.124E+01	**4.675E+00**	2.160E+00	8.748E+00	5.477E+01	6.944E+01	5.548E+01
	Best	**1.000E+00**	**1.000E+00**	1.030E+00	4.790E+02	2.530E+00	1.004E+00	**1.000E+00**	4.866E+00	**1.000E+00**	**1.000E+00**	4.866E+00
	Worst	2.946E+02	1.112E+03	2.278E+01	1.532E+03	7.459E+01	2.045E+01	**4.866E+00**	1.942E+01	4.290E+02	4.329E+02	2.984E+02
	Median	**1.000E+00**	2.254E+02	4.867E+00	1.015E+03	1.364E+01	1.016E+00	**1.000E+00**	4.866E+00	1.200E+00	2.933E+00	1.942E+01
	STD	5.267E+01	2.442E+02	7.087E+00	3.054E+02	1.978E+01	6.889E+00	**1.772E+00**	6.437E+00	1.216E+02	1.348E+02	9.983E+01
F8	Mean	**1.000E+00**	**1.000E+00**	1.001E+00	1.548E+00	1.010E+00	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**
	Best	**1.000E+00**	**1.000E+00**	**1.000E+00**	1.345E+00	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**
	Worst	**1.000E+00**	1.000E+00	1.008E+00	1.915E+00	1.046E+00	**1.000E+00**	**1.000E+00**	**1.000E+00**	1.001E+00	**1.000E+00**	**1.000E+00**
	Median	**1.000E+00**	**1.000E+00**	**1.000E+00**	1.554E+00	1.005E+00	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**	**1.000E+00**
	STD	**0.000E+00**	1.210E-12	1.471E-03	1.454E-01	1.186E-02	2.006E-06	**0.000E+00**	1.166E-13	1.465E-04	**0.000E+00**	5.992E-12
F9	Mean	**1.067E+00**	1.693E+00	2.322E+00	6.481E+02	1.041E+01	1.590E+00	1.337E+00	1.330E+00	1.919E+00	1.121E+00	1.231E+00
	Best	**1.030E+00**	1.192E+00	1.215E+00	2.696E+02	1.258E+00	1.227E+00	1.184E+00	1.205E+00	1.114E+00	1.060E+00	1.065E+00
	Worst	**1.135E+00**	2.496E+00	5.242E+00	1.147E+03	3.802E+01	2.538E+00	1.448E+00	1.442E+00	5.583E+00	1.232E+00	1.387E+00
	Median	**1.066E+00**	1.687E+00	1.664E+00	6.450E+02	7.163E+00	1.481E+00	1.334E+00	1.333E+00	1.459E+00	1.113E+00	1.222E+00
	STD	**2.572E-02**	3.470E-01	1.180E+00	1.882E+02	8.625E+00	2.776E-01	6.087E-02	6.252E-02	1.190E+00	4.590E-02	8.665E-02
F10	Mean	**1.499E+01**	2.115E+01	2.138E+01	2.186E+01	1.847E+01	2.102E+01	2.134E+01	2.147E+01	2.042E+01	1.684E+01	2.103E+01
	Best	**1.000E+00**	2.100E+01	2.120E+01	2.159E+01	2.192E+00	2.101E+01	2.114E+01	2.127E+01	3.596E+00	**1.000E+00**	2.100E+01
	Worst	**2.100E+01**	2.155E+01	2.156E+01	2.205E+01	2.149E+01	2.105E+01	2.147E+01	2.161E+01	**2.100E+01**	2.121E+01	2.114E+01
	Median	**2.098E+01**	2.106E+01	2.139E+01	2.188E+01	2.134E+01	2.102E+01	2.134E+01	2.148E+01	2.100E+01	2.100E+01	2.101E+01
	STD	9.156E+00	1.820E-01	9.164E-02	1.165E-01	6.128E+00	**9.719E-03**	7.280E-02	8.239E-02	3.124E+00	7.852E+00	4.236E-02
Friedman	Average Ranking	3.108696	4.521739	4.369565	10.69565	5.76087	6.586957	7.652174	8.217391	6.413043	3.652174	5.021739
	Ranking	1	4	3	11	6	8	9	10	7	2	5

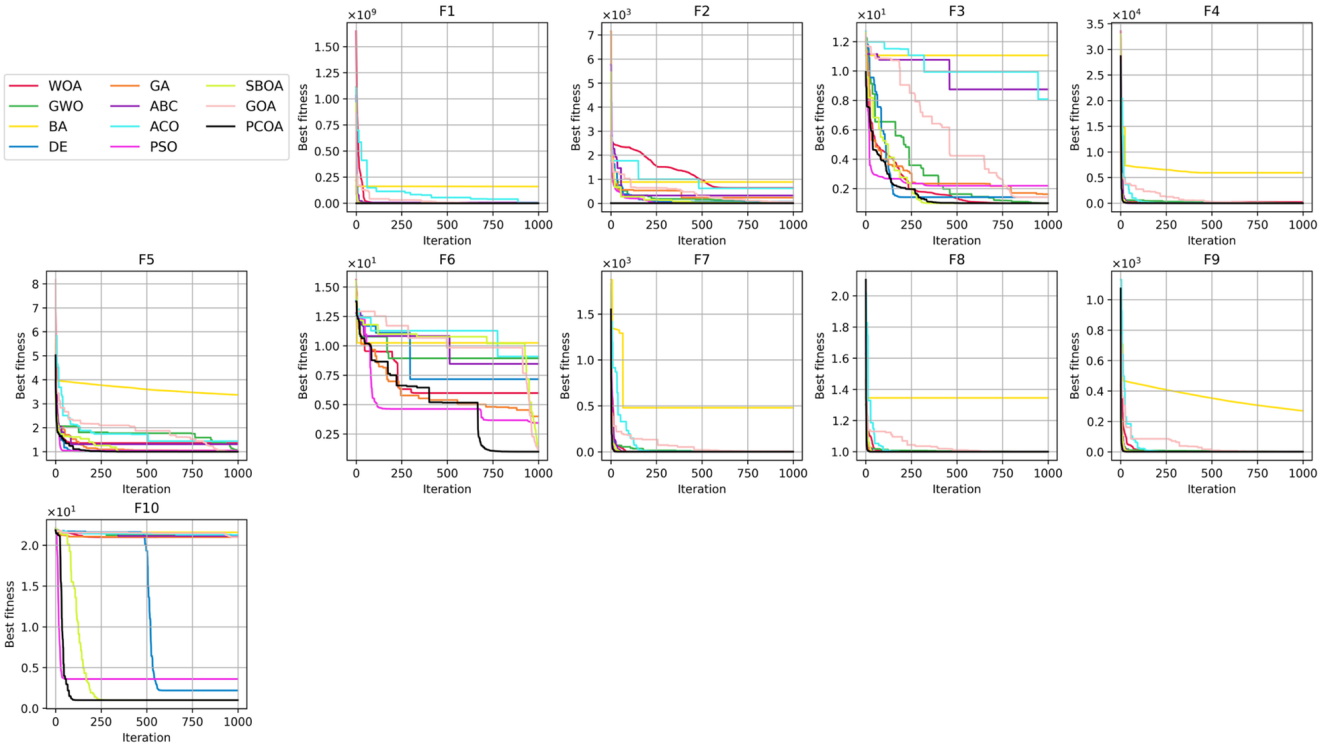

Figure 4. Convergence curves of PCOA and ten algorithms for CEC 2019 benchmark functions.

Table 5. The Wilcoxon rank-sum test results between PCOA and ten comparative algorithms on IEEE CEC 2019 Test Functions

F	WOA p (Sig.)	GWO p (Sig.)	BA p (Sig.)	DE p (Sig.)	GA p (Sig.)	ABC p (Sig.)	ACO p (Sig.)	PSO p (Sig.)	SBOA p (Sig.)	GOA p (Sig.)
F1	(+) 1.27E-10	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.66E-02	(+) 2.87E-11
F2	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11
F3	(+) 1.93E-08	(+) 2.11E-07	(+) 2.87E-11	(+) 7.76E-11	(+) 2.74E-10	(+) 2.87E-11	(+) 2.87E-11	(+) 3.18E-11	(+) 1.84E-04	(+) 1.27E-10
F4	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 3.06E-09	(+) 3.88E-11
F5	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 1.27E-10	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 3.51E-11	(+) 9.77E-04	(+) 3.01E-10
F6	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 4.78E-09	(+) 2.87E-11	(+) 2.87E-11	(+) 5.32E-10	(~) 8.13E-01	(~) 2.04E-01
F7	(+) 4.84E-10	(-) 1.49E-08	(+) 2.87E-11	(+) 4.00E-09	(-) 3.21E-08	(-) 6.78E-07	(-) 4.00E-09	(+) 3.21E-08	(+) 1.64E-05	(+) 1.62E-09
F8	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11	(+) 2.87E-11	(~) 1.00E+00	(+) 2.87E-11
F9	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 3.51E-11	(+) 1.93E-06	(+) 6.41E-10
F10	(+) 5.23E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 1.67E-06	(+) 2.87E-11	(+) 2.87E-11	(+) 2.87E-11	(+) 5.22E-09	(+) 1.05E-05	(+) 3.51E-11
[W | T | L]	[10 | 0 | 0]	[9 | 0 | 1]	[10 | 0 | 0]	[10 | 0 | 0]	[9 | 0 | 1]	[8 | 1 | 1]	[9 | 0 | 1]	[10 | 0 | 0]	[8 | 2 | 0]	[9 | 1 | 0]

shown as follows:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ru(1-u), \qquad 0 \le x \le 1, \qquad 0 \le t \le t_M, \tag{39}$$

with the initial condition:

$$u(x,0) = f(x) = \frac{1}{\left(1+e^{x/\sqrt{6}}\right)^2}, \qquad 0 \le x \le 1, \tag{40}$$

and the boundary conditions:

$$u(0,t) = p(t) = \frac{1}{\left(1+e^{-5/6t}\right)^2}, \qquad 0 \le t \le t_M, \tag{41}$$

$$u(1,t) = q(t), \qquad 0 \le t \le t_M, \tag{42}$$

in which $q(t)$ represents the unknown continuous boundary function as:

$$u(1,t) = q(t) = \frac{1}{\left(1+e^{1/\sqrt{6}-5/6t}\right)^2}, \tag{43}$$

and the over-specified condition:

$$u(a_0,t_j) = s(t_j), \tag{44}$$

and the exact solution of this problem is:

$$u(x,t) = \frac{1}{\left(1+e^{x/\sqrt{6}-5/6t}\right)^2}, \tag{45}$$

$$F(u(x,t)) = u(x,t)(1-u(x,t)), \tag{46}$$

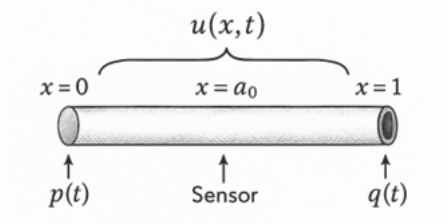

Figure 5. A 1-D Heat Conduction Problem.

In real world applications, it is common to employ a sensor placed at an interior point a_0 (i.e. $a_0 = 0.5$) to measure and collect observational data from a system governed by a partial differential equation (PDE). This system models heat conduction in a solid bar of unit length. As the boundary condition is unknown, the problem is treated as an inverse problem. The primary objective is to determine $q(t)$, the temperature at time t and at the endpoint $x = 1$, as well as to identify the unknown boundary condition. To compensate for the lack of boundary information, a sensor is placed within the interval $[a,b]$ to collect temperature data (Fig. 5).

This data is then used to estimate an unknown function within the system. To illustrate the significance of observed data, consider a classic example from heat transfer: a heat conduction problem described by a simple linear PDE.

3.1 Discretization Process

To construct the numerical solution, we define nodal points (x_i,t_j) over the domain $[0,1]\times[0,t_M]$, where x_i and t_j represent the discretized spatial and temporal coordinates, respectively. A schematic view of the discretized computational domain in both space and time is depicted in Figure 6, which helps visualize the nodal structure used in the fully implicit numerical formulation. Equation (39) is discretized using the fully implicit method. Consequently, the discretized form of the Fisher equation is derived as follows:

$$0 = x_1 < x_2 < \cdots < x_n = 1 = a, \qquad h = x_{i+1} - x_i = \frac{1}{N}, \tag{47}$$

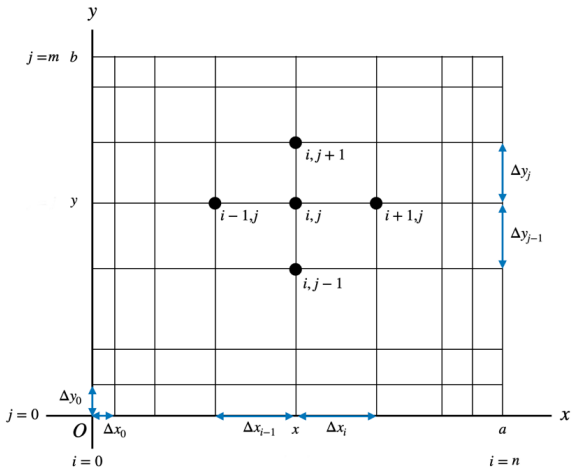

Figure 6. Schematic view of the space-time discretization used in the solution.

$$0 = t_0 < t_1 < \cdots < 1 = b \cdots < 1, \qquad k = t_{j+1} - t_j, \tag{48}$$

Now, by applying the fully implicit method to Equation (39), we obtain:

$$\frac{1}{k}(U_{i,j+1} - U_{i,j}) = \frac{1}{h^2}(U_{i+1,j+1} - 2U_{i,j+1} + U_{i-1,j+1}) + F(U_{i,j}), \tag{49}$$

where $U_{i,j} = U(x_i, t_j)$. Then, the Fisher equation can thus be expressed in the following form:

$$U_{i,j} + kF(U_{i,j}) = -rU_{i-1,j+1} + (1+2r)U_{i,j+1} - rU_{i+1,j+1}, \tag{50}$$

$$U_{i,0} = f(ih), \tag{51}$$

$$U_{0,j} = p(jk), \tag{52}$$

$$U_{1,j} = q(jk), \tag{53}$$

$$r = \frac{k}{h^2}, \tag{54}$$

which gives:

$$\begin{pmatrix} 1+2r & -r & 0 & \cdots & 0 & 0 & 0 \\ -r & 1+2r & -r & \cdots & 0 & 0 & 0 \\ 0 & -r & 1+2r & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2r & -r & 0 \\ 0 & 0 & 0 & \cdots & -r & 1+2r & -r \\ 0 & 0 & 0 & \cdots & 0 & -r & 1+2r \end{pmatrix}_{(N-1)\times(N-1)} \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N-2,j+1} \\ u_{N-1,j+1} \end{pmatrix}_{(N-1)} - rk \begin{pmatrix} p_{j+1} \\ 0 \\ \vdots \\ 0 \\ q_{j+1} \end{pmatrix}_{(N-1)}$$

$$= \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N-2,j} \\ u_{N-1,j} \end{pmatrix}_{(N-1)} + k \begin{pmatrix} F(u_{1,j}) \\ F(u_{2,j}) \\ \vdots \\ F(u_{N-2,j}) \\ F(u_{N-1,j}) \end{pmatrix}_{(N-1)}, \tag{55}$$

we obtain the following system:

$$AX = B, \tag{56}$$

where A is the coefficient matrix, X is the vector of unknowns, and B is the right-hand side vector. The corresponding components for A, X, and B are given as follows:

$$A = \begin{pmatrix} 1+2r & -r & 0 & \cdots & 0 & 0 & 0 \\ -r & 1+2r & -r & \cdots & 0 & 0 & 0 \\ 0 & -r & 1+2r & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2r & -r & 0 \\ 0 & 0 & 0 & \cdots & -r & 1+2r & -r \\ 0 & 0 & 0 & \cdots & 0 & -r & 1+2r \end{pmatrix}, \tag{57}$$

$$X = \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{N-2,j+1} \\ u_{N-1,j+1} \end{pmatrix}, \tag{58}$$

$$B = rk \begin{pmatrix} p_{j+1} \\ 0 \\ \vdots \\ 0 \\ q_{j+1} \end{pmatrix} + \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{N-2,j} \\ u_{N-1,j} \end{pmatrix} + k \begin{pmatrix} F(u_{1,j}) \\ F(u_{2,j}) \\ \vdots \\ F(u_{N-2,j}) \\ F(u_{N-1,j}) \end{pmatrix}, \tag{59}$$

By solving this linear system, $(N-1)$ unknown axial values along the boundary $x = 1$ are obtained.

To tackle inverse problems involving partial differential equations, it is necessary to first determine the unknown conditions. Once these conditions are known, the problem is converted into a direct problem, which can then be accurately solved using the outlined numerical methods. For the PCOA algorithm, the cost function is defined as follows and is designed to identify the optimal solution by minimizing the value of Equation (60):

$$S = \left[\frac{1}{m} \sum_{j=1}^{m} (U_{a_0,j} - s_j)^2 \right]^{1/2}, \tag{60}$$

where $U_{a_0,j}$ represents the computed solution at the observation points (i.e. $a_0 = 0.5$), s_j is the corresponding measured data, and m is the total number of observations. The objective is to minimize this cost to obtain the optimal solution.

To obtain a stable and accurate numerical solution for the Huxley equation, the fully implicit finite difference method is employed. This approach offers enhanced stability, particularly for stiff nonlinear reactiondiffusion problems, and is well-suited for long-time integration.

The generalized one-dimensional Huxley equation is given by:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku(1-u)(u-a), 0 \le x \le 1, 0 \le t \le t_M, \tag{61}$$

where $u(x,t)$ denotes the unknown function, D is the diffusion coefficient, k is the reaction rate constant, and $a \in (0,1)$ is a threshold parameter. The domain is discretized using a uniform grid with $x_i = ih$ for $i = 0,1,\ldots,N$, and $t_j = jk$ for $j = 0,1,\ldots,t_M$, where h and k denote the spatial and temporal step sizes, respectively.

The time derivative is approximated using a backward (implicit) difference:

$$\left.\frac{\partial u}{\partial t}\right|_{x_i,t_{j+1}} \approx \frac{u_{i,j+1} - u_{i,j}}{k}, \tag{62}$$

while the second-order spatial derivative is discretized using central differences evaluated at the new time level:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i,t_{j+1}} \approx \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2}. \tag{63}$$

The nonlinear reaction term is also evaluated at the implicit time level:

$$u(1-u)(u-a)|_{x_i,t_{j+1}} \approx u_{i,j+1}(1-u_{i,j+1})(u_{i,j+1}-a). \tag{64}$$

Substituting these approximations into the original PDE yields the fully discretized implicit form:

$$\frac{u_{i,j+1} - u_{i,j}}{k} = D\frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2} + k\,u_{i,j+1}(1-u_{i,j+1})(u_{i,j+1}-a). \tag{65}$$

Rewriting in matrix form leads to a nonlinear system $AX = B$, where A is a tridiagonal matrix derived from the diffusion term, and B incorporates known values from the previous time step and the nonlinear reaction term; with the initial condition:

$$U(x,0) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2\sqrt{2}}\right), 0 \le x \le 1, \tag{66}$$

and the boundary conditions:

$$u(0,t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{-t}{4}\right), 0 \le x \le t_M, \tag{67}$$

$$u(1,t) = q(t), 0 \le t \le t_M, \tag{68}$$

in which $q(t)$ represents the unknown continuous boundary function as:

$$u(1,t) = q(t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{1}{2\sqrt{2}}\left(1 - \frac{t}{\sqrt{2}}\right)\right), 0 \le x \le t_M, \tag{69}$$

and the over-specified condition:

$$u(a_0,t_j) = s(t_j), \tag{70}$$

and the exact solution of this problem is:

$$u(x,t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{1}{2\sqrt{2}}\left(x - \frac{t}{\sqrt{2}}\right)\right), \tag{71}$$

$$F(u(x,t)) = (1-u(x,t))(u(x,t)-1). \tag{72}$$

In inverse heat conduction problems (IHCPs), estimation accuracy is influenced by two fundamental error sources. The first is the deterministic error (or bias), which originates from model simplifications and discretization assumptions. The second is the stochastic error, caused by the amplification of measurement noise during the inversion process. The overall impact of these errors is typically evaluated through the root mean squared error (RMSE), which encapsulates both bias and variance contributions, thereby representing the total estimation error [10].

$$S = \left[\frac{1}{N-1}\sum_{i=1}^{N}(\hat{q}_i - q_i)^2\right]^{1/2}, \tag{73}$$

where N denotes the total number of estimated values, and $\hat{q}$ represents the predicted values obtained from the interpolated numerical solution, used in comparison with observed data to compute the error metric.

To provide a more comprehensive comparison and to evaluate the performance of the proposed PCOA algorithm, experiments were conducted on six different grid sizes, each corresponding to a distinct discretization level. Tables 6 and 7 present the RMSE results obtained for the Fisher and Huxley equations, respectively. The findings clearly demonstrate the robustness and effectiveness of PCOA across all tested discretization schemes for both parabolic PDEs.

Additionally, to enable a more detailed investigation, the over-specified condition was examined under two different settings: $a_0 = 0.5$ and $a_0 = 0.7$ except for the grid size 5×5, where the settings were adjusted to $a_0 = 0.4$ and $a_0 = 0.6$. This supplementary analysis highlights the adaptability of the algorithm under varying boundary information, reinforcing its reliability in solving inverse problems.

Moreover, the PCOA was evaluated against ten competitive algorithms including WOA, GWO, BA, DE, GA, ABC, ACO, PSO, SBOA, and GOA, all of which were also tested on standard benchmark functions. This comparative study further emphasizes the superior accuracy and stability of PCOA in addressing inverse parabolic PDEs.

To further assess the convergence behavior of the proposed method in comparison with other algorithms, the evolution of the estimated boundary function $q(t)$ at the spatial location $x = 1$ was analyzed for each method. The convergence profiles were individually plotted for all algorithms to illustrate their stability and accuracy in reconstructing the unknown boundary condition. These convergence plots are presented in Figures 7 through 14, corresponding to the Fisher and Huxley equations, respectively. As evidenced by the results presented in Tables 6 and 7, multiple spatiotemporal discretizations have been employed to solve the Fisher and Huxley equations. This comprehensive comparison clearly highlights that the most accurate solutions in this study were achieved using a grid size of 10×20 for the Fisher equation and a grid size of 10×10 for the Huxley equation.

Table 6. RMSE Results of the Estimated Unknown Boundary Function - Fisher Equation along with Execution Times

Grid Size	a_0	PCOA	WOA	GWO	BA	DE	GA	ABC	ACO	PSO	SBOA	GOA
5 × 5	0.4*	**4.54E-03**	1.25E-02	5.62E-03	2.53E-01	6.50E-03	9.27E-03	4.45E-03	7.74E-03	**4.54E-03**	**4.54E-03**	4.56E-03
	0.5*	**1.30E-03**	2.29E-03	1.33E-03	8.30E-02	9.19E-04	1.52E-03	1.33E-03	1.35E-03	1.35E-03	1.35E-03	1.34E-03
*Execution Time (s)		29.4	12	13.3	12.1	13	14.8	27.5	9.1	12.5	21	42.7
10 × 5	0.5*	**2.67E-03**	7.38E-03	2.78E-03	1.91E-01	4.51E-03	4.42E-03	2.86E-03	5.51E-03	2.86E-03	2.85E-03	2.84E-03
	0.7	**7.23E-04**	7.83E-04	8.00E-04	1.26E-01	6.18E-04	1.89E-03	8.00E-04	7.77E-04	8.00E-04	8.00E-04	8.00E-04
*Execution Time (s)		44.6	19.3	18.3	17.4	18.7	20.4	39.5	11.6	18.4	32.2	49.5
10 × 10	0.5*	**7.08E-03**	5.79E-02	1.15E-01	1.47E-01	1.15E-01	2.22E-01	4.11E-02	1.99E-01	4.38E-02	2.31E-02	9.52E-02
	0.7	**5.87E-04**	2.48E-02	3.91E-02	1.01E-01	3.01E-02	2.39E-02	6.22E-03	2.93E-03	3.47E-02	1.91E-03	3.10E-03
*Execution Time (s)		70	28.9	28.7	27.5	28.6	30.3	58.2	18.7	27.8	50.8	60
10 × 20	0.5*	**1.28E-02**	1.53E-01	2.77E-01	1.70E-01	1.56E-01	1.71E-01	1.86E-01	2.79E-01	8.38E-02	8.62E-02	2.37E-01
	0.7	**1.50E-03**	8.83E-02	3.27E-01	1.90E-01	2.37E-01	7.45E-02	1.33E-01	2.14E-01	6.22E-02	3.13E-02	1.39E-01
*Execution Time (s)		125	46.8	46.6	45.9	49.2	51.5	99	35.7	47.6	91	84
10 × 100	0.5*	**4.25E-02**	2.18E-01	2.63E-01	2.94E-01	2.39E-01	2.43E-01	3.17E-01	3.37E-01	2.39E-01	1.80E-01	3.06E-01
	0.7	**1.97E-02**	1.91E-01	2.59E-01	2.89E-01	1.84E-01	2.39E-01	3.03E-01	3.21E-01	2.19E-01	1.27E-01	2.28E-01
*Execution Time (s)		589	211	218	221	221	222	448	162	218	431	252
10 × 200	0.5*	**7.73E-02**	2.21E-01	3.21E-01	2.76E-01	2.08E-01	2.83E-01	3.34E-01	3.67E-01	2.30E-01	2.08E-01	3.42E-01
	0.7	**4.16E-02**	1.86E-01	2.97E-01	2.87E-01	1.96E-01	2.82E-01	3.05E-01	3.46E-01	2.34E-01	1.70E-01	3.09E-01
*Execution Time (s)		1182	435	435	432	433	437	885	318	436	862	467

3.2 Parameter Recovery Accuracy for Fisher and Huxley Equations

Accurate recovery of unknown or time-dependent parameters is a fundamental challenge in solving inverse problems. Table 8 presents representative values of the exact and estimated boundary function $q(t)$ at selected time points for the Fisher and Huxley equations, under over-specified conditions ($a_0 = 0.5$ and $a_0 = 0.7$). The corresponding percentage error is also reported. These values illustrate the high accuracy of the proposed algorithm in recovering the true parameter values over the interval $t \in [0.1, 1.0]$.

Table 7. RMSE Results of the Estimated Unknown Boundary Function - Huxley Equation along with Execution Times

Grid Size	a_0	PCOA	WOA	GWO	BA	DE	GA	ABC	ACO	PSO	SBOA	GOA
5 × 5	0.4*	**3.32E-03**	**3.32E-03**	**3.32E-03**	4.43E-02	**3.32E-03**	3.33E-03	**3.32E-03**	**3.32E-03**	**3.32E-03**	**3.32E-03**	**3.32E-03**
	0.6	**2.20E-03**	**2.20E-03**	**2.20E-03**	5.83E-02	**2.20E-03**	**2.22E-03**	**2.20E-03**	**2.20E-03**	**2.20E-03**	**2.20E-03**	**2.20E-03**
*Execution Time (s)		53.3	21.2	22.1	20.5	22.6	24.1	46.3	13.4	21.5	38.5	52.2
10 × 5	0.5*	**2.74E-03**	2.76E-03	2.75E-03	8.25E-02	2.77E-03	2.76E-03	2.76E-03	2.76E-03	2.76E-03	2.76E-03	2.76E-03
	0.7	**5.19E-04**	5.20E-04	**5.19E-04**	2.38E-02	**5.19E-04**	5.04E-04	5.20E-04	5.20E-04	5.20E-04	5.20E-04	5.20E-04
*Execution Time (s)		66	26.1	26.5	25.2	27.5	29.7	56.2	15.1	26	47.4	59.5
10 × 10	0.5*	**1.36E-03**	1.47E-03	**1.36E-03**	1.67E-01	7.45E-03	1.87E-03	1.79E-03	**1.36E-03**	3.48E-03	**1.36E-03**	**1.36E-03**
	0.7	**8.09E-04**	8.12E-04	8.12E-04	1.75E-01	3.09E-03	8.47E-04	8.12E-04	8.12E-04	8.16E-04	8.12E-04	8.12E-04
*Execution Time (s)		84	32.4	33.1	32.1	34.4	36.2	70	22.2	33.2	60	68
10 × 20	0.5*	**6.78E-04**	4.64E-02	1.97E-03	1.98E-01	7.30E-02	3.88E-02	1.19E-01	8.61E-02	5.52E-02	7.49E-04	6.80E-04
	0.7	**4.04E-04**	7.08E-03	4.57E-04	2.04E-01	2.84E-02	2.26E-02	4.61E-02	3.14E-02	4.29E-02	**4.04E-04**	**4.04E-04**
*Execution Time (s)		122	45.2	46.2	45.8	48.2	49.9	99	37.4	46.1	119	85
10 × 100	0.5*	**8.26E-02**	2.16E-01	2.39E-01	3.29E-01	1.81E-01	2.17E-01	2.49E-01	2.76E-01	2.25E-01	1.61E-01	2.56E-01
	0.7	**2.37E-02**	1.47E-01	2.05E-01	3.09E-01	1.51E-01	2.01E-01	2.30E-01	2.66E-01	2.46E-01	8.57E-02	1.48E-01
*Execution Time (s)		440	163	165	166	167	170	342	142	163	332	203
10 × 200	0.5*	**1.15E-01**	2.44E-01	2.83E-01	3.23E-01	1.98E-01	2.36E-01	2.64E-01	2.88E-01	2.45E-01	1.93E-01	3.21E-01
	0.7	**9.49E-02**	1.98E-01	2.60E-01	3.35E-01	1.87E-01	2.17E-01	2.32E-01	2.79E-01	2.66E-01	1.61E-01	2.90E-01
*Execution Time (s)		829	306	306	303	306	305	616	260	306	604	337

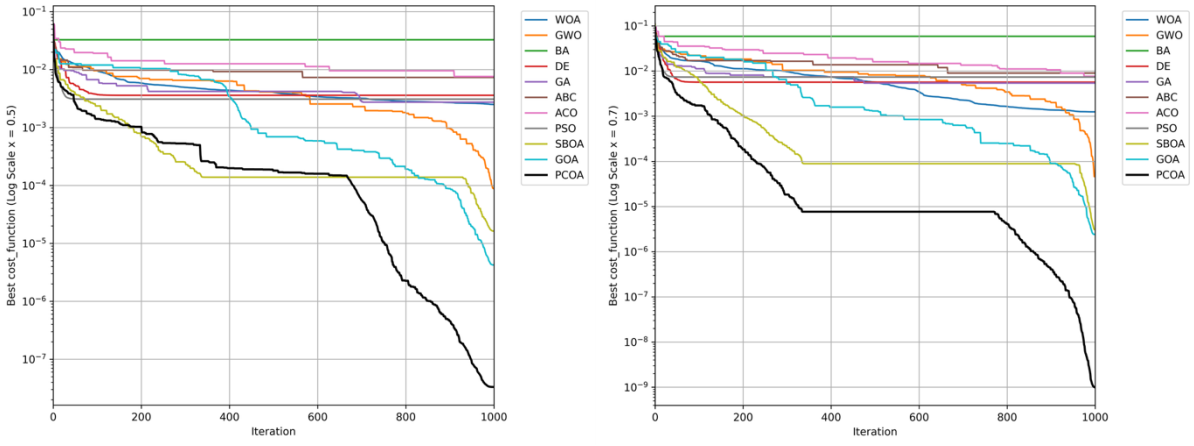

Figure 7. Best cost (fitness) plots for the Fisher equation at over-specified condition $x = 0.5$ and $x = 0.7$ at grid size 10×20, compared with ten competitive algorithms. The left plot shows the best fitness values computed at $x = 0.5$, while the right plot presents the corresponding values at $x = 0.7$ over 1000 iterations.

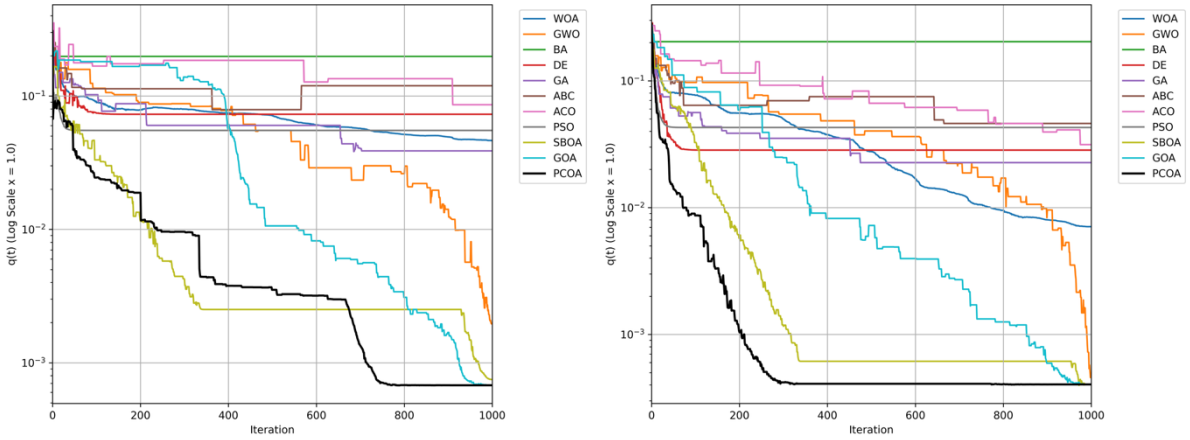

Figure 8. Convergence curves of the estimated boundary points $q(t)$ for the Fisher equation at grid size 10×20, compared with ten competitive algorithms. The left plot corresponds to the case with over-specified condition $a_0 = 0.5$, and the right plot to the case with $a_0 = 0.7$, both evaluated at the boundary point $x = 1$ over 1000 iterations.

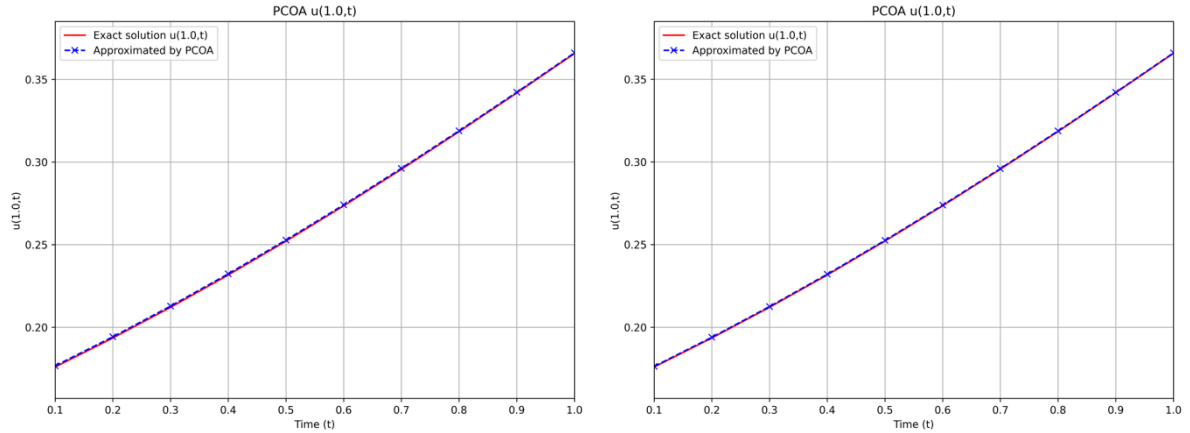

Figure 9. Exact and approximated boundary function $q(t)$ for the Fisher equation at grid size 10×20 by PCOA. The left plot corresponds to the case with the over-specified condition $a_0 = 0.5$, and the right plot to the case with $a_0 = 0.7$.

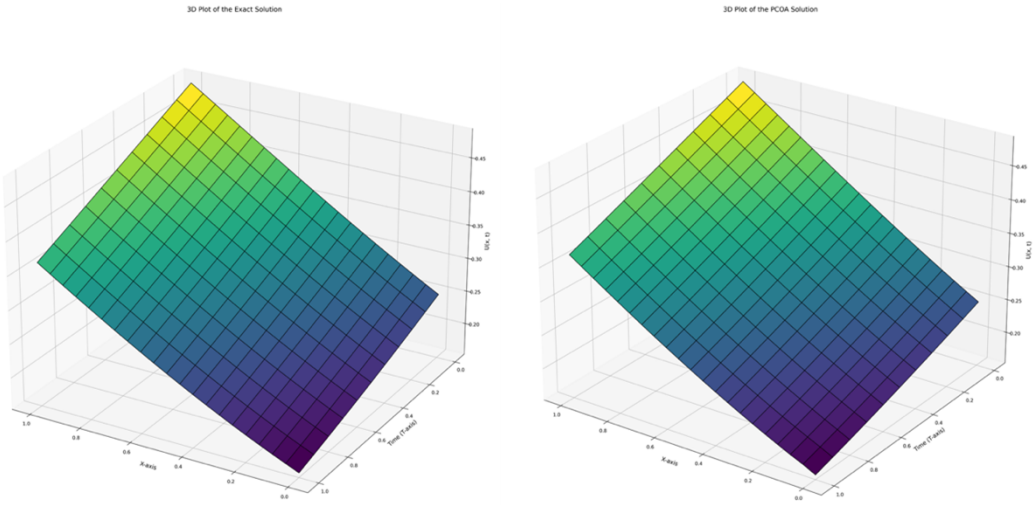

Figure 10. Exact and PCOA-estimated solution $u(x,t)$ for the Fisher equation. The left plot shows the exact solution, while the right plot presents the solution obtained using the PCOA algorithm.

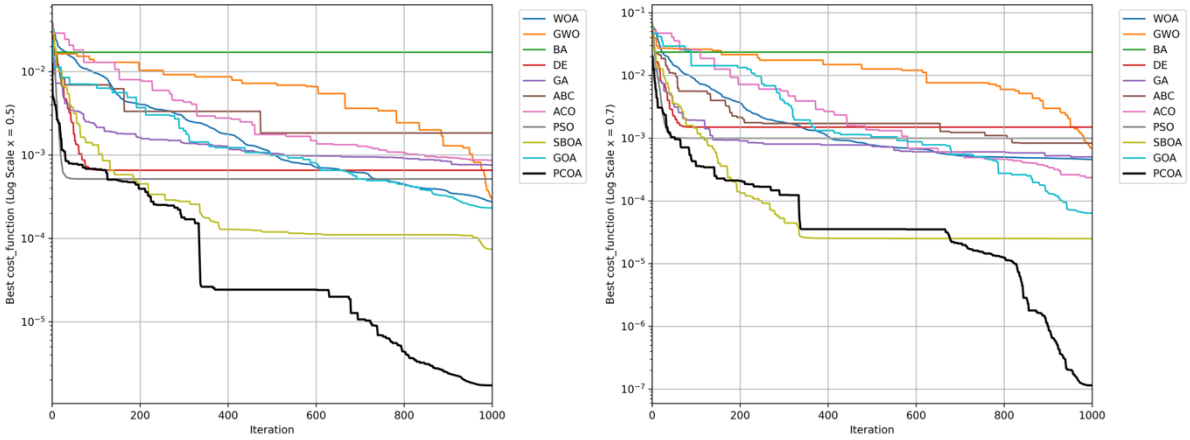

Figure 11. Best cost (fitness) plots for the Huxley equation at over-specified condition $x = 0.5$ and $x = 0.7$ at grid size 10×10, compared with ten competitive algorithms. The left plot shows the best fitness values computed at $x = 0.5$, while the right plot presents the corresponding values at $x = 0.7$ over 1000 iterations.

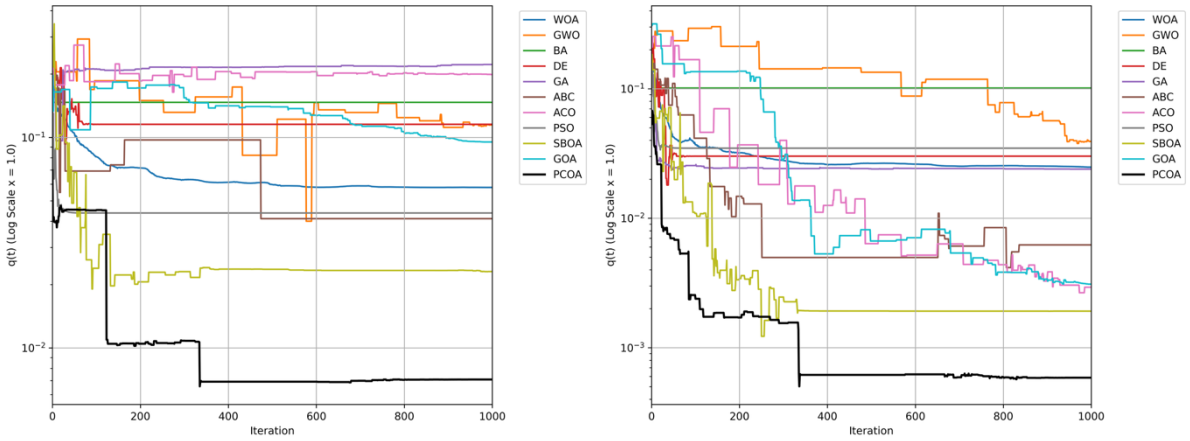

Figure 12. Convergence curves of the estimated boundary points $q(t)$ for the Huxley equation at grid size 10×10, compared with ten competitive algorithms. The left plot corresponds to the case with over-specified condition $a_0 = 0.5$, and the right plot to the case with $a_0 = 0.7$, both evaluated at the boundary point $x = 1$ over 1000 iterations.

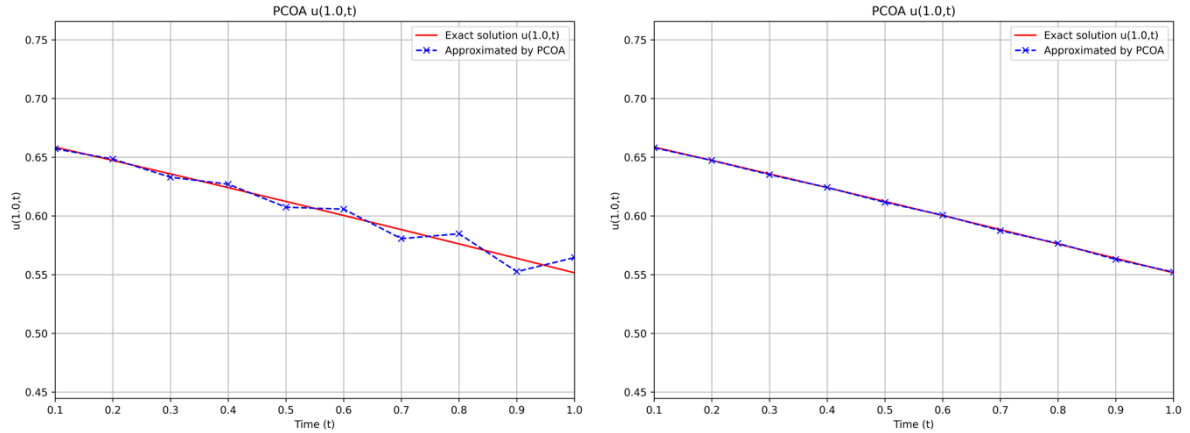

Figure 13. Exact and approximated boundary function $q(t)$ for the Huxley equation at grid size 10×10 by PCOA. The left plot corresponds to the case with the over-specified condition $a_0 = 0.5$, and the right plot to the case with $a_0 = 0.7$.

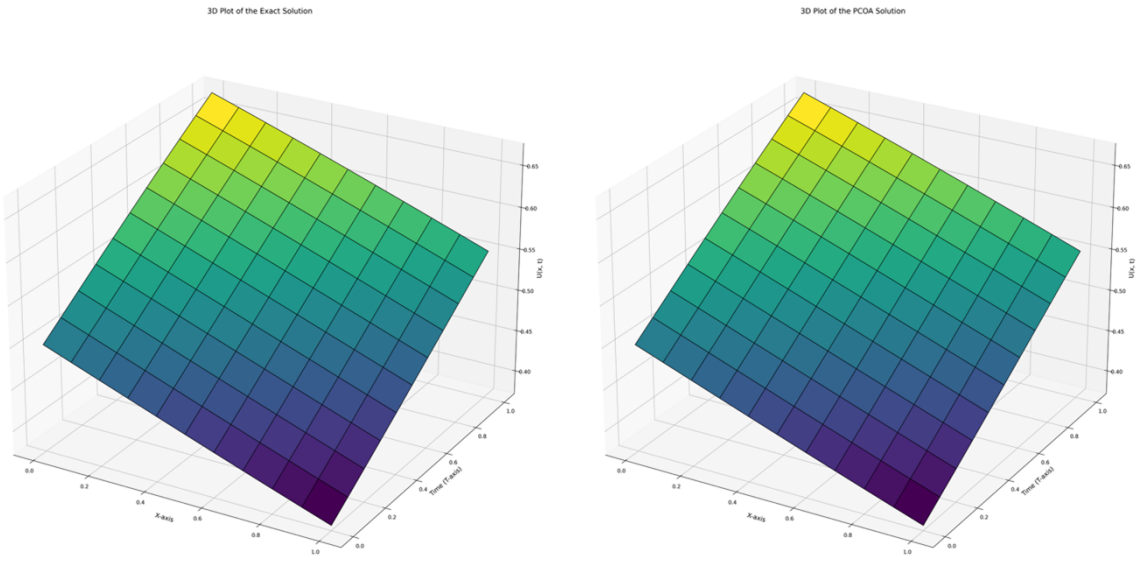

Figure 14. - Exact and PCOA-estimated solution $u(x,t)$ for the Fisher equation at grid size 10×10. The left plot shows the exact solution, while the right plot presents the solution obtained using the PCOA algorithm.

Table 8. Exact and estimated boundary function at selected time points for the Fisher and Huxley equations

Equation	t	a_0	Exact $q(t)$	Obtained $q(t)$	Error (%)
Fisher (10×20)	0.2	0.5	0.193509037	0.194261845	0.39
		0.7		0.193963647	0.23
	0.4	0.5	0.231630453	0.232357993	0.31
		0.7		0.232089148	0.20
	0.6	0.5	0.27344726	0.274105864	0.24
		0.7		0.273870965	0.15
	0.8	0.5	0.318375189	0.318924072	0.17
		0.7		0.318743525	0.12
	1	0.5	0.365661381	0.366070293	0.11
		0.7		0.365936679	0.08
Huxley (10×10)	0.2	0.5	0.647280538	0.648419279	0.18
		0.7		0.647273507	0.00
	0.4	0.5	0.624127992	0.627027887	0.46
		0.7		0.624249195	0.02
	0.6	0.5	0.600393937	0.605764862	0.89
		0.7		0.600662	0.04
	0.8	0.5	0.576178903	0.584844495	1.50
		0.7		0.576611031	0.07
	1	0.5	0.551592413	0.564516074	2.34
		0.7		0.552206441	0.11

4 Conclusion

The algorithms efficacy was rigorously validated through benchmark evaluations, including twenty-three classical functions and the CEC 2017 test suite. Comparative analysis against ten state-of-the-art algorithms (e.g., DE, PSO, GWO, GA, WOA, and others) demonstrated that PCOA consistently outperformed its peers in terms of convergence rate, robustness, and solution accuracy. Statistical tests such as Wilcoxon rank-sum and Friedman tests confirmed the significance of these performance improvements.

Beyond benchmark scenarios, PCOA was successfully applied to the inverse formulations of the Fisher and Huxley equations, showcasing its ability to estimate unknown boundary conditions with high precision, even in the absence of prior information or initial guesses. Numerical results confirmed that the algorithm remains effective across different discretization levels and over-specified boundary settings. Moreover, the method showed strong adaptability in solving both homogeneous and non-homogeneous NIPDEs, and proved compatible with established numerical solvers such as the CrankNicolson method.

In summary, PCOA not only offers a powerful and generalizable framework for solving a wide range of real-world and theoretical optimization problems but also introduces a new paradigm for algorithmic design through cinematic metaphor. Its demonstrated performance, conceptual novelty, and adaptability suggest strong potential for future extensions across high-dimensional, dynamic, and multi-objective optimization challenges.

5 Broader Implications and Future Research

By merging narrative-driven strategic reasoning with computational intelligence, this study establishes a transformative paradigm in metaheuristic optimization. The broader implications of this approach include:

1. Advanced Decision-Space Modeling: Cinematic-inspired frameworks enable more structured yet flexible navigation strategies for tackling high-dimensional, nonlinear search problems.

2. Enhanced Communication Mechanisms for Swarm Intelligence: Inspired by cooperative storytelling and cinematic fleet coordination, the proposed framework lays a foundation for sophisticated inter-agent collaboration models that enhance efficiency and scalability.

3. Implementing similar strategies on other goal-oriented movies can be very exciting.

4. The main idea, in addition to incorporating such innovations inspired by movies, can pave the way for drawing inspiration from video games as well.

Authors' Contributions

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

Data Availability

All data in the paper are available from the corresponding authors upon reasonable request.

Conflicts of Interest

The authors declare that there is no conflict of interest.

Ethical Considerations

The authors have diligently addressed ethical concerns, such as informed consent, plagiarism, data fabrication, misconduct, falsification, double publication, redundancy, submission, and other related matters.

Funding

This research did not receive any grant from funding agencies in the public, commercial, or nonprofit sectors.

Acknowledgments

We thank our colleagues and funding agencies for their support.

References

[1] C. Storey, Applications of a hill climbing method of optimization, Chem. Eng. Sci., 17(1), 45–52 (1962).

[2] V. Granville, J. P. Rasson, and M. Krivanek, Simulated annealing: a proof of convergence, IEEE Trans. Pattern Anal. Mach. Intell., 16(6), 652–656 (1994).

[3] J. A. Nelder and R. Mead, A simplex method for function minimization, Comput. J., 7(4), 308–313 (1965).

[4] S. M. Almufti, A. A. Shaban, Z. A. Ali, R. I. Ali, and J. A. Dela Fuente, Overview of metaheuristic algorithms, Polaris Global Journal of Scholarly Research and Trends, 2(2), 10–32 (2023).

[5] A. Aliyari Boroujeni, M. R. Ghaemi, and R. Pourgholi, Solving the transportation problem using meta-heuristic algorithms, Analytical and Numerical Solutions for Nonlinear Equations, 9(1), 12–19 (2025).

[6] S. W. Kareem, K. W. H. Ali, S. Askar, F. S. Xoshaba, and R. Hawezi, Metaheuristic algorithms in optimization and its application: a review, JAREE, 6(1) (2022).

[7] J. H. Holland, Genetic algorithms, Sci. Am., 267(1), 66–72 (1992).

[8] M. Dorigo, M. Birattari, and T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag., 1(4), 28–39 (2006).

[9] J. Kennedy and R. Eberhart, Particle swarm optimization, Proceedings of ICNN95 - International Conference on Neural Networks, 4, 1942–1948 (1995).

[10] A. Aliyari Boroujeni, R. Pourgholi, and S. H. Tabasi, A new improved teaching-learning-based optimization (ITLBO) algorithm for solving nonlinear inverse partial differential equation problems, Comput. Appl. Math., 42(2) (2023).

[11] A. Aliyari Boroujeni and A. Khadivar, Solving the inverse Fisher problem using a discretized teaching-learning-based optimization algorithm, Analytical and Numerical Solutions for Nonlinear Equations, 10(1), 35–49 (2025).

[12] B. Brahim, M. Kobayashi, M. Al Ali, T. Khatir, and M. E. A. E. Elmeliani, Metaheuristic optimization algorithms: an overview, HCMCOU Journal of Science Advances in Computational Structures (2024).

[13] A. Mishra and L. Goel, Metaheuristic algorithms in smart farming: an analytical survey, IETE Technical Review, 41(1), 46–65 (2024).

[14] M. Abdeyazdan, S. M. Dehno, and S. H. Tarighinejad, An investigation and comparison of invasive weed, flower pollination and krill evolutionary algorithms, Int. J. Adv. Comput. Sci. Appl., 7(7) (2016).

[15] A. Akgul, Y. Karaca, M. A. L. I. Pala, M. E. Cimen, A. L. I. F. Boz, and M. Z. Yildiz, Chaos theory, advanced metaheuristic algorithms and their deep learning architecture optimization applications: a review, 32(3) (2024).

[16] A. Kaveh, N. G. Malek, A. D. Eslamlou, and M. Azimi, An open-source framework for the FE modeling and optimal design of fiber-steered variable-stiffness composite cylinders using water strider algorithm, Mech. Based Des. Struct. Mach., 51(1), 138–158 (2023).

[17] J. Dreo et al., Paradiseo: from a modular framework for evolutionary computation to the automated design of metaheuristics, GECCO 2021 Companion, 1522–1530 (2021).

[18] F. Peres and M. Castelli, Combinatorial optimization problems and metaheuristics: review, challenges, design, and development, Appl. Sci., 11(14), 6449 (2021).

[19] A. A. Boroujeni, R. Pourgholi, and S. H. Tabasi, Solving inverse partial differential equations problems by using teaching learning based optimization algorithm.

[20] M. Baghel, S. Agrawal, and S. Silakari, Survey of metaheuristic algorithms for combinatorial optimization, Int. J. Comput. Appl., 58(19), 21–31 (2012).

[21] Z. Li, V. Tam, and L. K. Yeung, An adaptive multi-population optimization algorithm for global continuous optimization, IEEE Access, 9, 19960–19989 (2021).

[22] S. C. Chu, H. C. Huang, J. F. Roddick, and J. S. Pan, Overview of algorithms for swarm intelligence, Lecture Notes in Computer Science, 6922, 28–41 (2011).

[23] D. Martens, B. Baesens, and T. Fawcett, Editorial survey: swarm intelligence for data mining, Mach. Learn., 82(1), 1–42 (2011).

[24] Y. Qawqzeh, M. T. Alharbi, A. Jaradat, and K. N. A. Sattar, A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments, PeerJ Comput. Sci., 7, e696 (2021).

[25] P. S. Mann and S. Singh, Improved artificial bee colony metaheuristic for energy-efficient clustering in wireless sensor networks, Artif. Intell. Rev., 51(3), 329–354 (2019).

[26] M. Mavrovouniotis, S. Yang, M. Van, C. Li, and M. Polycarpou, Ant colony optimization algorithms for dynamic optimization: a case study of the dynamic travelling salesperson problem, IEEE Comput. Intell. Mag., 15(1), 52–63 (2020).

[27] A. Abraham, H. Guo, and H. Liu, Swarm intelligence: foundations, perspectives and applications, Stud. Comput. Intell., 26, 3–25 (2006).

[28] R. Garcia-Rodenas, L. J. Linares, and J. A. Lopez-Gomez, A memetic chaotic gravitational search algorithm for unconstrained global optimization problems, Appl. Soft Comput., 79, 14–29 (2019).

[29] K. Hussain, W. Zhu, and M. N. Mohd Salleh, Long-term memory Harris hawk optimization for high dimensional and optimal power flow problems, IEEE Access, 7, 147596–147616 (2019).

[30] A. A. Boroujeni and A. Khadivar, Solving the traveling salesman problem using a modified teaching-learning-based optimization algorithm, Int. J. Ind. Eng. Prod. Res., 36(2), 170–184 (2025).

[31] A. A. Boroujeni, R. Pourgholi, and S. H. Tabasi, Solving inverse partial differential equations problems by using teaching learning based optimization algorithm, 738–755 (2024).

[32] A. Aliyari Boroujeni, R. Pourgholi, and S. H. Tabasi, Numerical solutions of KdV and mKdV equations: using sequence and multi-core parallelization implementation, J. Comput. Appl. Math., 454, 116184 (2025).

[33] R. Storn and K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim., 11, 341–359 (1997).

[34] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev., 42(1), 21–57 (2014).

[35] X.-S. Yang, A new metaheuristic bat-inspired algorithm.

[36] S. Mirjalili and A. Lewis, The whale optimization algorithm, Adv. Eng. Softw., 95, 51–67 (2016).

[37] S. Mirjalili, S. M. Mirjalili, and A. Lewis, Grey wolf optimizer, Adv. Eng. Softw., 69, 46–61 (2014).

[38] Y. Fu, D. Liu, J. Chen, and L. He, Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems, Artif. Intell. Rev., 57(5) (2024).

[39] S. Saremi, S. Mirjalili, and A. Lewis, Grasshopper optimisation algorithm: theory and application, Adv. Eng. Softw., 105, 30–47 (2017).

[40] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness, and A. H. Gandomi, Aquila optimizer: a novel meta-heuristic optimization algorithm, Comput. Ind. Eng., 157, 107250 (2021).